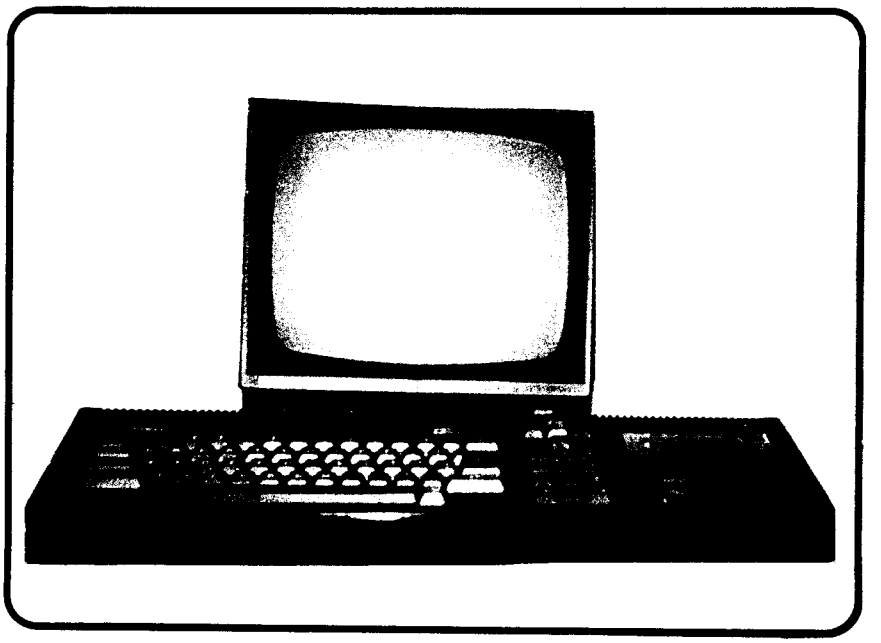
# PRINT—OUT

Price 70p

## *ISSUE SIX*



By :- Thomas Defoe, Mark Gearing and Jonathan Haddock

Contributors :- Bob Taylor, Alan Scully & R. Sergeant

***Including:*** ARTIST PROGRAM
TEARAWAY REVIEWED
CPM PLUS
ADVANCED M/C

# Index

## Miscellaneous



## Features



## Reviews



## Programming



---


We would like to express our thanks to Mr. Gearing and Black Horse Agencies Januarys for the continued use of their photocopier in producing Print-Out. Please note that we do not support piracy, unless back-ups are for the sole use of the original owner (in which case it is very sensible see opposite).

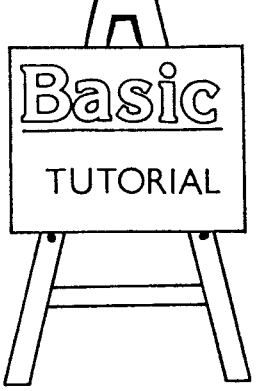
# Editorial

WELCOME TO ISSUE SIX OF PRINT-OUT !!!

This issue of the magazine that you are holding at the moment was very nearly never produced. However this near disaster was not caused by a shortage of funds, the technical failure of the photocopier, a faded printer ribbon, the Gulf Crisis, the authors' desertion (would we ever!), a world paper shortage or the recent heatwave!! No, just as our publication date loomed on the horizon & the last articles were hurriedly being typed into Protext to soon be printed out and given their beautiful (!) artwork, disaster struck. Imbecile that I am, in an attempt to make a back-up copy of the Prospell dictionary, I accidentally FORMATTED the data disc with all of this issue's files on it. My raised voice could be heard throughout Bishop's Stortford, as I realised that we hadn't made a single back-up of any of the files (fools that we are). Desperation set in when I found out that all of the programs had also been wiped off. Several phone calls to our contributors later (our thanks go to Alan Scully, Richard Sergeant and especially Bob Taylor for contributing in this issue), we had retrieved about a third of the magazine. We then started on the very long task of re-typing all of the articles in this issue & trying to rewrite all of the programs. When we had at last finished, checking to label all the discs and making sure that the write-protect holes were open we decided that the motto of this true (& painful) story is to always keep lots of working BACK-UPS!!

If we haven't managed to blank too many more discs our birthday issue will be completed by about 30th September. Further details of some of its contents can be found on page 23 & copies can be ordered in advance (see the Order Form on page 40-41). If you have any questions or problems concerning the CPC, please write to the address below and we'll do our best to answer it. The address to write to is :-

PRINT-OUT, 8 Maze Green Road,
Bishop's Stortford, Herts CM23 2PJ.




# BEGINNER'S BASIC

## PART SIX

Starting from Issue Seven, there are going to be some major changes in the way that the BASIC sections are organised. As from next issue, Beginner's BASIC will be covering all sorts of topics to do with BASIC on the CPC, by looking at various new commands in turn & explaining how to use them. Advanced BASIC will, meanwhile, take a slight detour for a few issues by examining at such things as Variables, Arrays, Strings and Tokens in the CPC's memory. There will also be a regular (I hope!) section containing useful subroutines & other tips to do with BASIC programming.

With luck this will allow us to cover a larger number of different aspects of BASIC without losing any of the other articles in the magazine but if you've any comments or thoughts on these ideas, why not fill in the questionnaire from Issue Seven ?

Before delving deeper into any new BASIC commands, next issue, we ought to remind ourselves of everything that has been covered so far in this series, and also to tidy up one or two lose ends.

If I were to mention 'separators' in BASIC, most people wouldn't know what I was talking about, & yet the truth is that everyone has used them hundreds of times. Even the simple line `10 PRINT "Hello";name$:REM Hello` contains several examples of separators or 'delimiters' as they are sometimes known. All of the 'spaces' are delimiters as is the colon. Their purpose is to split lines up into chunks which can be recognised by the CPC. While this is not hugely important, it is the sort of term which often crops up in computer books and it's nice (and helpful) to understand what is going on.

Many of the commands that we have looked, were explained in their simplest form. What I mean by this, is that these commands could have optional sections added onto them. If you take a very obvious command, such as LIST, you will see that when it was introduced, we just looked at as the command LIST on its own. However, there are many other forms of list all of which can have numbers after the actual command. Take these examples :-

| | | |
|---|---|---|
| LIST 50 | - | This will just display (list) line 50 of a program |
| LIST 50-100 | - | This displays all the lines with numbers of between 50 & 100 |
| LIST -100 | - | This lists all the lines with numbers upto 100 |
| LIST 100- | - | This lists all the lines with numbers greater than 100 |

When you add onto this, the ability to add a stream expression, you'll see that even a command as apparently simple as LIST can instantly become very much more complex. The reason for mentioning this is twofold. Firstly, to illustrate that it is often worth experimenting with commands (with the help of the manual) and secondly to explain why we will be going back over some old commands in future issues.

We have already looked at variables in BASIC and have seen how fundamental they are to a good program. As you will have read, there are two kinds of variable, the string variable and the numerical variable. In Issue Two, I mentioned that a string variable must have a dollar sign ($) as its last character, but a numerical variable must not. However, numerical variables may be sub-divided in two sections:

1) Integer variables - Here the last character must be a percent sign (%). The difference between this & any other numerical variable, is that any decimal places are discarded (ie the value may only be a whole number, an integer). All integer variables can have values of between -32768 and 32767 and their advantage is that instructions which use integer variables will run faster because no decimal calculations have to be done. Before you can use these, you have to use a DEFINT (DEFine INTeger command) of the form DEFINT A,G,D

2) Real variables - Here the the last character is an exclamation mark (!) and these allow numbers to have decimal places or not. All numerical variables default (ie start off) as real variables. Before being used, you need to issue a DEFREAL command (DEFine REAL) such as DEFREAL J,K,L

This short program will illustrate the differences between real and integer.

```
10 DEFINT a
20 a%=1.5
30 PRINT a%
40 DEFREAL a
50 a!=1.5
60 PRINT a!
```

For completeness's sake, you can also define an integer to be a string by using DEFSTR (eg DEFSTR d).
Once an integer has been defined in this way, you no longer need to include the variable type marker at the end of the variable. Type in this program

```
10 DEFINT a
20 a=45.5:print a
30 DEFREAL a
40 a=45.5:print a
50 DEFSTR a
60 a="Hello":print a
```

Note that even the string variables don't need a dollar sign after them because the computer already knows that they are strings (as they've been defined previously). It is also possible to define a large number of variables. For example DEFINT a-z will define all of the letters from a through to z as integer variables. In general, it is best to steer clear of these commands and not to worry about them, but just to use normal string (with a $ sign) & numerical variables (without anything after it) as these are the least complicated to use.

We have looked as GOTO as part of a program (in order to create a loop)...

```
10 PRINT "Hello";
20 GOTO 10
```

....but you can use it as a direct command. A direct command is an instruction, or a series of instructions, which do not have a line number at the beginning & are therefore executed as soon as ENTER (or RETURN on a 6128) is pressed.

If you had a long program which included a section to INPUT someone's name and you wanted to test just this final part you could do so using GOTO or RUN. Say, for example, that your program looked something like this :-

```
 10 REM A very long program which goes on until
 20 REM line 900 doing all sorts of things. But
 30 REM line 900 is the start of the INPUTting
890 REM routine and this is where we want to go to
900 INPUT "What is your name ";name$
910 PRINT "Goodbye ";name$
920 END
```

In order to execute just lines 900 onwards, simply use GOTO 900. You could even use RUN 900 and, as you can see, this is another example of an optional section in a command (up until now we had only met it as just RUN).

That's it for this issue, but in Issue Seven we will be using some of the more interesting and more complex (it goes without saying!) commands available to the CPC user.

Until then, have a brief look through your user manual, in particular the 'Keywords' section, and experiment with some of the commands that interest you. If you don't understand some of the terms, compare it with a command that we've looked at in this series. If it looks very complex, it probably contains plenty of optional sections - so start by missing these out and then gradually include them when you are confident with the rest of the command.

---

# SMALL ADS

REVENGE OF CHAOS now available on disc for the CPC featuring 76 locations and graphics. Also comes with free game - Alien Planet and costs £3.95. Also available on disc:- Lords of Magic £3.95, Island of Chaos £3.95. Cheques Postal Orders to: T.Kingsmill, 202 Park Street, St Albans, Herts AL2 2AQ.

FREE ADVENTURES, Text-only and Quilled. Jason and the Argonauts,Dungeon,Roog, Welladay,Spaced-Out,Doomlords (Parts 1,2,3),Firestone,Can I Cheat Death? Just send tape/disc, SAE and 50p per game. Simon Avery, Marden Farm, Old Exeter Road, Chudleigh, South Devon TQ13 0DR.


# NEWS

from ALAN SCULLY

Hello, good evening, and welcome to another round-up of Scull PD software! Unfortunately, due to the fact that very few people were ordering P.D. on tape, the library now does only disk PD. I know it is a shame, but when you receive upto 20 orders per day and when only one of them is for cassette PD, you just cannot justify the time spent copying and updating the tape library. However, it does mean a better service for the majority of our Public Domain customers.

You may remember that in the previous issue of Print-Out, two of my games, Rebound and Bandit, were reviewed. I would like to remind you that these NON-PD games are still available. To receive them both send £5 for disk or £3 for tape. Full details are included in the Stock-List Disk-Magazine (details below).

Now onto the PD....Simon "Romeo Squirrel" Avery has written some excellent adventures using the Quill and he has donated them to my library. Two of these, Jason and the Argonauts and Firestone are reviewed elsewhere in this issue. The other adventures that he has sent are Doomlords (in 3 parts),Can I Cheat Death, Spacy, Dungeon adventure, Adult 2, Welladay and Roog. Thanks Simon!

The latest games to enter the library are MAC II and MAC III & these games were written by Dino especially for Scull PD and they are brilliant! MAC 2 is played on a flick screen maze & features big multicolour graphics, ten levels, a level designer, 300 screens in total & an adjustable number of lives. Skulls (Sculls?) roam about the playing area and touching them is fatal. On screen, a map of the current level is shown and (if you are a sucker for punishment!) you can switch it off and play blind! Although the game is difficult, it is very addictive and you can expect to be glued to your keyboard for weeks!

The 13 mazes in MAC III are only a screen in size and although the graphics are smaller than those in MAC II, they are much more varied. The object of the game is to collect three objects within a time limit but only one appears at a time. Sounds easy? Well perhaps I'd better tell you about Mac's tendency to bounce of walls! Every time you reach a wall Mac bounces back the way he came. This means that to complete each level you must have lightening fast reactions.

On the next page, is the concise stock list as on August 1st 1990 (the size of this list is expected to triple by issue 7). If you wish to know exactly which programs are on the disks then send me a 3" disk + an S.S.A.J.B. (Stamped Self Addressed Jiffy Bag!) In return, you will be sent the STOCK-LIST DISK-MAGAZINE which contains the full stock list (with a mini-description of every program), full reviews of selected programs, colour screens shots from selected programs, tips & cheats on PD programs & information on the library. This disk is updated daily, so you can be sure of up-to-the-minute information.

# THE DISKS

PDD 01 - Serious 1 - Features 63 programs inc Linechek, and 2 Video titlers
PDD 02 - Games 1 - 31 games including several by myself
PDD 03 - Games 2 - 11 Mega games inc Mac I, Sugar Disk, Woboll, and Humpback
PDD 04 - Animation 1 - 4 pure machine coded animation demos for 128K
PDD 05 - DW Disk 1 - Features the amazing ST DESKTOP as raved in Amstrad Action
PDD 06 - DW Disk 2 - With MINICAD, MINICAD+, wordsearch designer, and more
PDD 07 - Applications 1 - Text Editors, Databases, Spread sheets & PageMaker
PDD 08 - Games 3 - 17 games including the TOTAL ECLIPSE DEMO from INCENTIVE
PDD 09 - Serious 2 - 6128 disk copier, fast formaters, new char sets, and more
PDD 10 - Games 4 - Two massive 6128 games and 3 adventures (1 features grafix)
PDD 11 - AI/Educational/Gfx - Eliza, Time Teacher, SFS, and 32 grafix progs
PDD 12 - Games 5 - Inc UNTOUCHABLES DEMO from OCEAN & FORCE FIELD (40K+ 6128)
PDD 13 - Animations 2 - 3 more 6128 animations using ANIMATOR by GREMLIN
PDD 14 - Art Disk 1 - 18 compressed pictures with un-compress option
PDD 15 - Adventures 1 - CAN I CHEAT DEATH (over 16's by Simon Avery), RAIDER (role playing), & BLUE RAIDER (2 parts)
PDD 16 - Geno - 2 part adventure with loadsa documents and PROTEXT DEMO
PDD 17 - Applications 2 - Wales tourist guide (as raved about in NCE), PROTEXT DEMO from ARNOR, Disk Library, Databases, and More
PDD 18 - Games 6 - 32 games inc Bruce Lee, Boxing, and many more
PDD 19 - Adventures 2 - All by SIMON AVERY - Adult 2, Spacy, Doomlords, 2 more
PDD 20 - CPM Misc 1 - 2 adventures (BASE,ISLAND), NewSweep, MFT, StarTrek, more
PDD 21 - Serious 3 - 57 progs including fonts, disk RSX's, Cat #8, and more
PDD 22 - Quiz Disk 1 - 29K IQ Quiz, Give us a Break! with 60K questions, & more
PDD 23 - CPM Applications 1 - Andybase, VDE word processor, SuperZap, and more
PDD 24 - CPM Applications 2 - NULU, Spell Checker, MaxelCat (disk Lib), & more
PDD 25 - CPM Applications 3 - Scrivener (Relational Database/Spreadsheet), UKM7
PDD 26 - Games 7 - Features MAC II, MAC III, Trumps, Solitaire, Darts, and more

---

All that remains is to give you my address and a few library details...

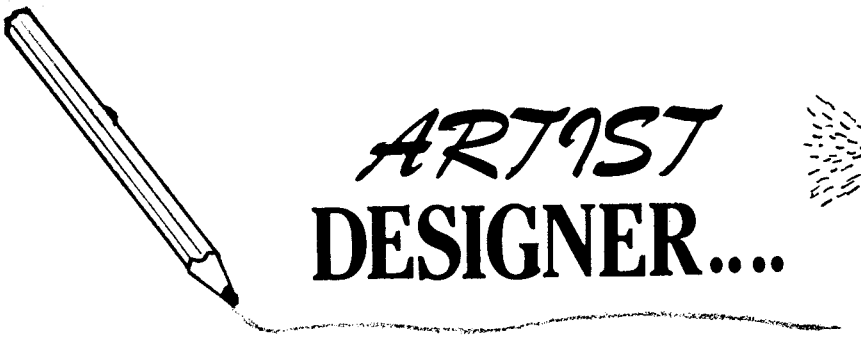
(1) To receive a disk full of PD (any TWO selections from the above list), send a disk, £1, and an SSAJB (see above). For the disk, either send a 3" disk or two 5 1/4" disks (on the 5 1/4" disks, only one side per disk is used in DATA format). Please note: all PD is supplied on an as-is basis.

(2) To receive the STOCK-LIST DISK-MAGAZINE (see above) send a disk + an SSAJB. For the disk, either send a 3" disk, or a 5 1/4" disk.

(3) If you have any PD from other PD libraries then send it on disk with an SSAJB. Disks will be returned with PD of your choice from the above list. Again, you can send either a 3" disk or two 5 1/4" disks (formatted in DATA format NOT 800K)

(4) The address is 119 Laurel Drive, East Kilbride, Glasgow G75 9JG, or you can telephone Alan on (03552) 24795 for a chat. Please phone Monday to Friday between 6pm and 11pm. All orders are dispatched within 24 hours.

(5) By the way, the library has the biggest range of PD software in the UK, and is the first to offer PD on 5 1/4" disk.



# ARTIST DESIGNER....

PROGRAM

This issue's program extravaganza is a fully menu-driven art package which runs in MODE 2. It features the ability to draw circles and boxes automatically (and also an erase box option), a shaded fill routine, print text (upto 6 times normal MODE 1 lettering size!!), spray can with variable spray density, pencil, eraser, movable line drawing facilities, save & load screen and a host of other options.

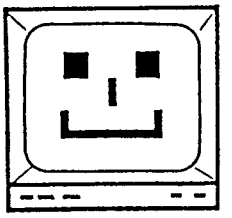
It is rather large and so you may find it helpful to know that the program (and a simple demo screen) are contained on this issue's program tape and disc, see page 40 for more details. But printed below are the instructions for how to use the 'Artist Designer'. Before running, it is very important to turn off any expansion ROMs that have been fitted apart from the disc drive ROM (if fitted). When run, the program will ask you what to call the screen that you may want to save but there's an option in the main program to change the filename. REMEMBER that the load/save options work on whatever the present filename is. After this has been done, you are entered into the screen designer proper.

Near the centre of the screen you will see a block, this is the cursor. If this block disappears at any time, this signals that the computer is doing some calculations or something else. The cursor can be moved around the screen using the cursor keys. NOTE: You are not allowed to move into the status display near the bottom of the screen. There are 3 options available to you when you begin:

1) Press CTRL-U (CTRL key and U at the same time). This is an undo feature which allows you to UNDO the last thing that you did only. Do not press it until you have done something otherwise it will blank the screen.
2) Press CTRL-C (CTRL key and C at the same time). This changes the coordinates that are shown in the bottom status line to the cursor's present coordinates. This display does not change automatically when you move.
3) Press CTRL-G (CTRL key and G at the same time).This allows you to enter a new set of X and Y coordinates in the bottom status line (press ENTER after each one) and the cursor then moves to those coordinates.

As well as these, you can move the cursor into the top menu. When the cursor is positioned over the name of the menu you wish to 'OPEN' press the [COPY] key. A menu will now unfold with a black arrow next to the top entry. The arrow can be moved using the up and down cursor keys. Pressing [COPY] will select the option that the arrow is pointing at. For example move to the IMPLEMENTS menu and then press [COPY], now move the arrow to LINES and press [COPY].You will be returned to the position you started off from. Now move the cursor somewhere on the main screen (not in the top menu) and press [COPY]. The block will disappear, if you now press the cursor keys, you'll see a line extending from the point where you pressed [COPY], this line can be moved anywhere on the screen. When the line is where you want it, press [COPY] again and you are free to draw another line, or choose another option.

All the options operate in a similar way, and all are selected/begun/ended by pressing [COPY]. The only time you have to press [ENTER] is when the program is waiting for you to enter a number or some text - [ENTER] then terminates the entry routine. Finally, it is possible to design additional filling patterns, & as the details are too lengthy to go in here, they're printed in a leaflet that goes with the program tape/disc. It's possible to do some fairly complex things with the fill routine, but if you do try any non-standard filling (ie filling a line with a speckle, or filling letters with lines), we advice you to save your screen first as it is possible to crash the computer with illegal fills. All of the patterns supplied if used to fill a normal area (no matter how complicated) will not cause a crash. An option for a screen dump to a printed is provided in the listing, but as there are so many printers available we have left this part blank for you to add a screen dump suitable for your printer (details on how to do this are given in the program cassette/disc leaflet. Have fun.



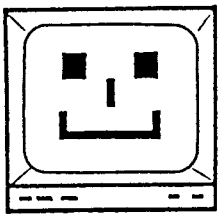
```
[B5] 10 REM   Artist Designer v1.0
[78] 20 REM     (c) 1990, T. Defoe
[7F] 30 INK 0,0:INK 1,24:PEN 1:PAPER 0:MODE 2
[44] 40 PRINT CHR$(23)+CHR$(1):BORDER 6:MOVE 0,0
[D5] 50 DRAW 0,399,1:DRAW 639,399:DRAW 639,0:DRAW 0,0
[3C] 60 LOCATE 31,8:PRINT "ART DESIGNER v1.0"
[2B] 70 LOCATE 32,10:PRINT "by Thomas Defoe"
[7F] 80 LOCATE 18,14:PRINT "Please wait whilst Machine Code is installed."
[19] 90 addr=&9000:RESTORE 100
[1D] 100 DATA 21,00,C0,01,00,40,11,00,40,ED,B0,C9,21,00,40,11,00,C0,01,00,40,ED
     ,B0,C9,end
[45] 110 READ a$:IF a$="end" THEN GOTO 140
[8C] 120 a=VAL("&"+a$):POKE addr,a
[88] 130 addr=addr+1:GOTO 110
[C6] 140 GOSUB 3940
[93] 150 LOCATE 16,16:INPUT "Please enter filename to save picture as : ",file$
[94] 160 IF LEN(file$)=0 OR LEN(file$)>8 THEN LOCATE 16,16:PRINT SPACE$(63):
     GOTO 150
[6F] 170 file$=UPPER$(file$)
[A9] 180 LOCATE 24,20:PRINT "Please press any key to continue"
[60] 190 a$=INKEY$:IF a$="" THEN 190
[6C] 200 GOSUB 1430 : REM Set up main screen
[95] 210 GOSUB 1570 : REM Move the cursor about
[60] 220 IF y<390 THEN GOTO 360 : REM Goto drawing/tool section
[98] 230 REM Option select from top bar
[8E] 240 opt=0:IF x<107 THEN opt=1
[D7] 250 IF x>107 AND x<214 THEN opt=2
[D2] 260 IF x>214 AND x<321 THEN opt=3
[94] 270 IF x>321 AND x<428 THEN opt=4
[FC] 280 IF x>428 AND x<535 THEN opt=5
[7C] 290 IF x>535 AND x<639 THEN opt=6
[A9] 300 IF opt=0 THEN GOTO 210
[BD] 310 TAG:PLOT 0,399,1:MOVE x,y:PRINT CHR$(232);:TAGOFF
[25] 320 CALL &9000 : REM Preserve screen for later
[96] 330 ON opt GOSUB 1960,2440,3110,3350,3550,3660
[D6] 340 IF opt=1 THEN 350 ELSE CALL &900C
[14] 350 x=320:y=300:GOSUB 2370:PLOT 0,399,1:TAG:MOVE x,y:PRINT CHR$(232);:TAGOFF:
     GOTO 210
```

```
[2C] 360 REM The actual drawing part of the program
[B5] 370 IF tool$="POINTER" THEN GOTO 210
[CD] 380 CALL &9000:storex=x:storey=y
[B8] 390 know$="PENCIL  LINES   ERASER  SPRAY   BOX
       ES   CIRCLES ERASEBOXTEXT    FILLING "
[58] 400 where=INSTR(know$,tool$)
[7C] 410 IF where=0 THEN 440
[C6] 420 where=INT(where/8+1):GOSUB 3830
[00] 430 ON where GOSUB 480,530,570,650,740,790,980,
       1100,1390
[6F] 440 PRINT #6,CHR$(23)+CHR$(1)
[30] 450 TAG:MOVE x,y:PRINT CHR$(232);:TAGOFF
[24] 460 FOR i=1 TO 40:NEXT i
[02] 470 GOTO 210
[11] 480 REM Pencil drawing
[5D] 490 PRINT #6,CHR$(23)+CHR$(0)
[DD] 500 GOSUB 3870
[10] 510 IF ret=1 THEN RETURN
[63] 520 PLOT x,y,1:ox=x:oy=y:GOTO 500
[90] 530 REM Line drawing routine
[E5] 540 GOSUB 3870
[18] 550 IF ret=1 THEN RETURN
[F4] 560 MOVE ox,oy:DRAW sx,sy:DRAW x,y:ox=x:oy=y:GOTO
       540
[02] 570 REM Rubber routine
[5C] 580 PRINT #6,CHR$(23)+CHR$(0)
[EF] 590 GOSUB 3870
[8E] 600 IF ret=1 THEN PRINT #6,CHR$(23)+CHR$(1):PLOT
       0,399,1:RETURN
[9A] 610 FOR i=x TO x+8
[FE] 620 MOVE i,y:DRAW i,y-8,0
[B2] 630 NEXT i
[80] 640 ox=x:oy=y:GOTO 590
[FF] 650 REM Spray can Routine
[59] 660 PRINT #6,CHR$(23)+CHR$(0)
[EC] 670 GOSUB 3870
[1F] 680 IF ret=1 THEN RETURN
[AC] 690 FOR i=1 TO dens
[B0] 700 px=INT(RND*8):py=INT(RND*8)
[48] 710 plus=INT(RND):IF plus=0 THEN px=px*-1 ELSE py
       =py*-1
[F6] 720 PLOT x+px,y+py,1:NEXT i
[69] 730 ox=x:oy=y:GOTO 670
[D6] 740 REM Box drawing routine
[E9] 750 GOSUB 3870
[1C] 760 IF ret=1 THEN RETURN
[E7] 770 MOVE sx,sy:DRAW ox,sy:DRAW ox,oy:DRAW sx,oy:
       DRAW sx,sy
[2C] 780 MOVE sx,sy:DRAW x,sy:DRAW x,y:DRAW sx,y:DRAW
       sx,sy:ox=x:oy=y:GOTO 750
[07] 790 REM Circle drawing routine
[FB] 800 IF INKEY(8)<>-1 AND x>sx THEN x=x-1
[0C] 810 IF INKEY(1)<>-1 AND x<620 THEN x=x+1
[69] 820 IF INKEY(9)<>-1 THEN GOTO 880
[10] 830 IF x=ox AND y=oy THEN GOTO 800
[B1] 840 dist=x-sx:odist=ox-sx
[8C] 850 MOVE sx-odist,sy:DRAW sx+odist,sy:MOVE sx,sy-
       odist:DRAW sx,sy+odist
```

# Linechecker

## A PROGRAM TYPING AID

All programs in Print-Out have Linecheck codes which are enclosed in brackets at the start of a line. Don't enter them in as they're designed to be used with Linechecker to eliminate errors when typing in programs which appear in this magazine. Please note, all programs will run whether Linechecker is being used or not. For information on how to use Linechecker, please see Issue Three.



```
[1C] 860 MOVE sx-dist,sy:DRAW sx+dist,sy:MOVE sx,sy-di
       st:DRAW sx,sy+dist
[07] 870 ox=x:oy=y:GOTO 800
[20] 880 MOVE sx-dist,sy:DRAW sx+dist,sy:MOVE sx,sy-di
       st:DRAW sx,sy+dist
[61] 890 PRINT #6,CHR$(23)+CHR$(0)
[C0] 900 IF sx=x THEN GOTO 950
[8F] 910 MOVE sx+dist*COS(0),sy+dist*SIN(0)
[E4] 920 DEG:FOR a=1 TO 370 STEP 5
[C0] 930 DRAW sx+dist*COS(a),sy+dist*SIN(a)
[67] 940 NEXT a
[D1] 950 GOSUB 3850
[7B] 960 PRINT #6,CHR$(23)+CHR$(1)
[D1] 970 RETURN
[A3] 980 REM Erase box
[D4] 990 GOSUB 740
[BB] 1000 MOVE sx,sy:DRAW x,sy:DRAW x,y:DRAW sx,y:DRAW
       sx,sy
[41] 1010 PRINT #6,CHR$(23)+CHR$(0)
[8D] 1020 IF y<sy THEN GOSUB 1040 ELSE GOSUB 1070
[5B] 1030 PRINT #6,CHR$(23)+CHR$(1):PLOT 0,399,1:RETURN
[67] 1040 FOR i=sy TO y STEP -1
[AD] 1050 MOVE sx,i:DRAW x,i,0
[53] 1060 NEXT i:RETURN
[73] 1070 FOR i=sy TO y
[B6] 1080 MOVE sx,i:DRAW x,i,0
[5C] 1090 NEXT i:RETURN
[40] 1100 PRINT #6,CHR$(23)+CHR$(0)
[91] 1110 LOCATE #3,1,2:PRINT #3,text$
[46] 1120 sx=x:sy=y:lin=sy
[EE] 1130 b=leth/2
```

```
[4C] 1140 num=LEN(text$)*8
[D8] 1150 FOR i=15 TO 0 STEP -2
[E6] 1160 FOR k=1 TO leth
[5C] 1170 FOR j=1 TO num
[69] 1180 IF TEST(j,i)=0 THEN GOSUB 1290 ELSE GOSUB 1340
[17] 1190 NEXT j
[CC] 1200 lin=lin-1:x=sx
[0C] 1210 NEXT k
[15] 1220 x=sx:lin=lin-1
[FC] 1230 NEXT i
[6B] 1240 PRINT #6,CHR$(23)+CHR$(1)
[61] 1250 PLOT 0,399,1
[26] 1260 x=sx:y=sy:ox=x:oy=y
[B7] 1270 GOSUB 2370
[77] 1280 RETURN
[35] 1290 FOR m=1 TO b*4
[71] 1300 PLOT x+m,lin,1
[24] 1310 NEXT m
[CB] 1320 x=x+(b*4)
[6A] 1330 RETURN
[2B] 1340 FOR m=1 TO b*4
[6D] 1350 PLOT x+m,lin,0
[33] 1360 NEXT m
[DA] 1370 x=x+(b*4)
[79] 1380 RETURN
[AF] 1390 REM Pattern fill
[08] 1400 PRINT CHR$(6)+CHR$(0)
[69] 1410 CALL &9040,wid,high,@shad$
[D4] 1420 PRINT CHR$(23)+CHR$(1):PLOT 0,399,1:RETURN
[BD] 1430 REM Set up the main screen
[F5] 1440 gpen=24:gpaper=0:x=320:y=300:tool$="POINTER"
[41] 1450 MODE 2:leth=1:MOVE 0,0:DRAW 0,399,1:MOVE 639
         ,0:DRAW 639,399:MOVE 0,18:DRAW 639,18
[6E] 1460 shad$=whole$:high=1:wid=1
[97] 1470 INK 1,gpen:INK 0,gpaper:dens=2
[59] 1480 WINDOW #6,80,80,25,25:PAPER #6,1:PEN #6,0
[A8] 1490 WINDOW #3,1,80,25,25:PAPER #3,1:PEN #3,0:CLS #3
[A8] 1500 GOSUB 2370
[2F] 1510 WINDOW #1,1,80,1,1:PAPER #1,1:PEN #1,0:CLS #1
[9C] 1520 PRINT #1," LOAD/SAVE      OPTIONS      COLOU
         RS    CHARACTERS     EXTRAS     IMPLEMENTS"
[61] 1530 PLOT 0,399,1
[71] 1540 PRINT #6,CHR$(23)+CHR$(1)
[C6] 1550 TAG:MOVE x,y:PRINT CHR$(232);
[4B] 1560 TAGOFF:RETURN
[CD] 1570 REM Move the cursor about - POINTER
[26] 1580 flag=0:ox=x:oy=y
[B4] 1590 IF INKEY(8)<>-1 AND x>9 THEN x=x-2
[D8] 1600 IF INKEY(1)<>-1 AND x<620 THEN x=x+2
[C7] 1610 IF INKEY(0)<>-1 AND y<391 THEN y=y+2
[0E] 1620 IF INKEY(2)<>-1 AND y>30 THEN y=y-2
[BB] 1630 IF INKEY(42)=128 THEN CALL &900C:x=storex:y=
         storey:GOTO 1580
[37] 1640 IF INKEY(62)=128 THEN LOCATE #3,23,1:PRINT
         #3,"X:";x:LOCATE #3,33,1:PRINT #3,"Y:";y;
[0F] 1650 IF INKEY(52)=128 THEN GOSUB 1850:GOTO 1580
[9A] 1660 IF INKEY(9)<>-1 THEN flag=1
[AA] 1670 IF flag=1 THEN 1690
[01] 1680 IF x=ox AND y=oy THEN GOTO 1590
[EF] 1690 TAG:MOVE ox,oy
[88] 1700 PRINT CHR$(232);
[79] 1710 MOVE x,y:PRINT CHR$(232);:TAGOFF
[4F] 1720 IF flag=1 THEN RETURN
[C1] 1730 GOTO 1580
[9C] 1740 REM Move pointer in sub-menu
[41] 1750 MOVE lx,382:TAG:PRINT CHR$(242);:TAGOFF
[84] 1760 my=382:om=my
[92] 1770 IF INKEY(0)<>-1 AND my<382 THEN my=my+16
[9A] 1780 IF INKEY(2)<>-1 AND my>ly THEN my=my-16
[FB] 1790 IF INKEY(9)<>-1 THEN RETURN
[FA] 1800 IF my=om THEN 1770
[B0] 1810 MOVE lx,my:TAG
[D2] 1820 PRINT CHR$(242);:MOVE lx,om:PRINT CHR$(242);
         :TAGOFF
[E8] 1830 GOSUB 3850
[0B] 1840 om=my:GOTO 1770
[82] 1850 REM Move coordinates
[08] 1860 TAG:MOVE ox,oy:PRINT CHR$(232);:TAGOFF
[F4] 1870 GOSUB 3850
[3F] 1880 LOCATE #3,23,1:PRINT #3,"       "
[C5] 1890 LOCATE #3,23,1:INPUT #3,"X";x
[B2] 1900 LOCATE #3,23,1:PRINT #3,"X:";x
[3C] 1910 LOCATE #3,33,1:PRINT #3,"       "
[03] 1920 LOCATE #3,33,1:INPUT #3,"Y";y
[0E] 1930 LOCATE #3,33,1:PRINT #3,"Y:";y
[67] 1940 TAG:MOVE x,y:PRINT CHR$(232);:TAGOFF
[7C] 1950 RETURN
[AC] 1960 REM LOAD/SAVE menu
[70] 1970 MOVE 0,380:DRAW 0,315,1:DRAW 104,315:DRAW 104
         ,380
[FE] 1980 WINDOW #2,2,12,2,5:PAPER #2,1:PEN #2,0:CLS #2
[EB] 1990 PAPER #1,0:PEN #1,1:LOCATE #1,1,1:PRINT #1,"
          LOAD/SAVE  "
[22] 2000 PAPER #1,1:PEN #1,0:PRINT #2,"LOAD file"
[77] 2010 PRINT #2,"SAVE file"
[AE] 2020 PRINT #2,"NEW file"
[06] 2030 PRINT #2,"EXIT MENU";
[2C] 2040 lx=86:ly=334:GOSUB 1740 : REM Move pointer in
         menu (bit of screen)
[D6] 2050 IF my>381 AND my<399 THEN mopt=1
[55] 2060 IF my>365 AND my<382 THEN mopt=2
[CF] 2070 IF my>349 AND my<366 THEN mopt=3
[F7] 2080 IF my<349 THEN mopt=4
[1F] 2090 IF mopt=4 THEN RETURN
[B5] 2100 ON mopt GOSUB 2120,2190,2270
[61] 2110 RETURN
[0E] 2120 REM Load file
[D5] 2130 CLS #2:PRINT #2,"LOAD (Y/N)"
[BE] 2140 a$=UPPER$(INKEY$):IF a$="" THEN 2140
[93] 2150 IF a$="N" THEN RETURN
[F7] 2160 IF a$<>"Y" THEN 2140
[9E] 2170 LOAD file$,&C000
[76] 2180 RETURN
[D8] 2190 REM Save file
[4E] 2200 CLS #2:PRINT #2,"SAVE (Y/N)"
[66] 2210 a$=UPPER$(INKEY$):IF a$="" THEN 2210
[BC] 2220 IF a$="N" THEN RETURN
[BF] 2230 IF a$<>"Y" THEN 2210
[22] 2240 CALL &900C
[BD] 2250 SAVE file$,b,&C000,&4000
[72] 2260 RETURN
[66] 2270 REM Change filename
[EC] 2280 GOSUB 3850
[9A] 2290 CLS #2:PRINT #2,"Enter file:"
[F0] 2300 LOCATE #2,1,2
[D0] 2310 INPUT #2,"",file$
[5C] 2320 IF LEN(file$)=0 THEN 2290
[12] 2330 IF LEN(file$)>8 THEN 2290
```

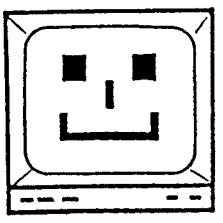
```
[C4] 2340 z=8-LEN(file$):FOR i=1 TO z:file$=file$+" ":
          NEXT i
[DD] 2350 file$=UPPER$(file$)
[74] 2360 RETURN
[36] 2370 LOCATE #3,1,1:PRINT #3,"Paper:";gpaper
[06] 2380 LOCATE #3,13,1:PRINT #3,"Pen:";gpen
[C2] 2390 LOCATE #3,23,1:PRINT #3,"X:";x
[FC] 2400 LOCATE #3,33,1:PRINT #3,"Y:";y
[74] 2410 LOCATE #3,43,1:PRINT #3,"Tool: ";tool$
[F7] 2420 LOCATE #3,62,1:PRINT #3,"Filename: ";file$
[6D] 2430 RETURN
[C6] 2440 REM The Options Menu
[6B] 2450 MOVE 108,380:DRAW 108,300,1:DRAW 242,300:DRAW
          242,380
[4B] 2460 WINDOW #2,15,30,2,6:PAPER #2,1:PEN #2,0:CLS #2
[51] 2470 PAPER #1,0:PEN #1,1:LOCATE #1,15,1:PRINT #1,"
            OPTIONS  "
[E7] 2480 PAPER #1,1:PEN #1,0:PRINT #2,"ERASE picture"
[E3] 2490 PRINT #2,"ERASE box":PRINT #2,"PRINT screen"
[DB] 2500 PRINT #2,"SPRAY density":PRINT #2,"EXIT menu"
[D0] 2510 lx=224:ly=318:GOSUB 1740
[14] 2520 IF my>381 AND my<399 THEN RUN
[EB] 2530 IF my>365 AND my<382 THEN tool$="ERASEBOX"
[8C] 2540 IF my>349 AND my<366 THEN GOSUB 2610
[49] 2550 IF my>333 AND my<350 THEN GOSUB 2570
[78] 2560 RETURN
[B6] 2570 CLS #2:PRINT #2,"Select Density"
[D5] 2580 INPUT #2,"(1-4) ",dens
[39] 2590 IF dens<1 OR dens>4 THEN GOTO 2570
[68] 2600 RETURN
[FD] 3110 REM Alter the screen colours
[2D] 3120 MOVE 220,380:DRAW 220,330,1:DRAW 316,330:DRAW
          316,380
[15] 3130 WINDOW #2,29,39,2,4:PAPER #2,1:PEN #2,0:CLS #2
[46] 3140 LOCATE 1,16
[90] 3150 PAPER #1,0:PEN #1,1:LOCATE #1,29,1:PRINT #1,"
            COLOURS  "
[15] 3160 PAPER #1,1:PEN #1,0:PRINT #2,"NEW paper"
[22] 3170 PRINT #2,"NEW pen":PRINT #2,"EXIT menu";
[82] 3180 lx=298:ly=357:GOSUB 1740
[E5] 3190 IF my>381 AND my<399 THEN mopt=1
[48] 3200 IF my>365 AND my<382 THEN mopt=2
[DB] 3210 IF my<365 THEN RETURN
[C3] 3220 IF mopt=1 THEN GOSUB 3250 ELSE GOSUB 3300
[92] 3230 GOSUB 1450
[6D] 3240 RETURN
[22] 3250 REM Change paper
[E7] 3260 GOSUB 3850
[BC] 3270 CLS #2:INPUT #2,"Choose new paper :",gpaper
[96] 3280 IF gpaper<0 OR gpaper>26 THEN 3270
 7 C] 3290 RETURN [
[B9] 3300 REM Change pen
[DA] 3310 GOSUB 3850
[FC] 3320 CLS #2:INPUT #2,"Choose new pen :",gpen
[00] 3330 IF gpen<0 OR gpen>26 THEN 3320
[6F] 3340 RETURN
[73] 3350 REM Character Menu
[B8] 3360 MOVE 324,380:DRAW 324,332,1:DRAW 442,332:DRAW
          442,380
[A4] 3370 WINDOW #2,42,55,2,4:PAPER #2,1:PEN #2,0:CLS #2
[10] 3380 PAPER #1,0:PEN #1,1:LOCATE #1,41,1:PRINT #1,"
            CHARACTERS "
[B8] 3390 PAPER #1,1:PEN #1,0:PRINT #2,"ENTER text"
[50] 3400 PRINT #2,"SELECT size"
[75] 3410 PRINT #2,"EXIT menu"
[98] 3420 lx=424:ly=350:GOSUB 1740
[7F] 3430 IF my>381 AND my<399 THEN GOSUB 3460
[38] 3440 IF my>365 AND my<382 THEN GOSUB 3510
[74] 3450 RETURN
[9A] 3460 WHILE INKEY$<>"":WEND:FOR i=1 TO 60:NEXT i:
          CLS #2:INPUT #3,"Enter text (no more than 40
           letters):",text$
[BB] 3470 IF LEN(text$)>40 THEN 3460
[C0] 3480 GOSUB 2370
[9C] 3490 tool$="TEXT    "
[67] 3500 RETURN
[44] 3510 CLS #2:PRINT #2,"Select size"
[A0] 3520 INPUT #2,"(1-6)";leth
[CA] 3530 IF leth<1 OR leth>6 THEN 3510
[73] 3540 RETURN
[37] 3550 REM Extras menu
[DA] 3560 MOVE 428,380:DRAW 428,315,1:DRAW 530,315:DRAW
          530,380
[2A] 3570 WINDOW #2,55,66,2,5:PAPER #2,1:PEN #2,0:CLS #2
[58] 3580 PAPER #1,0:PEN #1,1:LOCATE #1,55,1:PRINT #1,"
            EXTRAS   "
[A0] 3590 PAPER #1,1:PEN #1,0:PRINT #2,"CIRCLES"
[46] 3600 PRINT #2,"BOXES":PRINT #2,"SHADING":PRINT #2,
          "EXIT MENU"
[A8] 3610 lx=512:ly=334:GOSUB 1740
[1A] 3620 IF my>381 AND my<399 THEN tool$="CIRCLES "
[1E] 3630 IF my>365 AND my<382 THEN tool$="BOXES   "
[02] 3640 IF my>349 AND my<366 THEN GOSUB 4710
[78] 3650 RETURN
[D5] 3660 REM Select implement
[E0] 3670 MOVE 532,380:DRAW 532,284,1:DRAW 635,284:DRAW
          635,380
[F1] 3680 WINDOW #2,68,79,2,7:PAPER #2,1:PEN #2,0:CLS #2
[58] 3690 PAPER #1,0:PEN #1,1:LOCATE #1,68,1:PRINT #1,"
            IMPLEMENTS "
[A9] 3700 PAPER #1,1:PEN #1,0:PRINT #2,"PENCIL"
[5F] 3710 PRINT #2,"ERASER"
[B5] 3720 PRINT #2,"FILLING"
[F1] 3730 PRINT #2,"LINES"
[63] 3740 PRINT #2,"SPRAY"
[1B] 3750 PRINT #2,"EXIT MENU";
[B4] 3760 lx=617:ly=303:GOSUB 1740
[67] 3770 IF my>381 AND my<399 THEN tool$="PENCIL  "
[15] 3780 IF my>365 AND my<382 THEN tool$="ERASER  "
[BF] 3790 IF my>349 AND my<366 THEN tool$="FILLING "
[48] 3800 IF my>333 AND my<350 THEN tool$="LINES   "
[D1] 3810 IF my>317 AND my<334 THEN tool$="SPRAY   "
[73] 3820 RETURN
[64] 3830 TAG:MOVE x,y:PRINT CHR$(232);:TAGOFF
[1D] 3840 ox=x:oy=y:sx=x:sy=y
[94] 3850 WHILE INKEY$<>"":WEND
[BE] 3860 FOR i=1 TO 40:NEXT i:RETURN
[1C] 3870 ret=0
[92] 3880 IF INKEY(8)<>-1 AND x>9 THEN x=x-1
[D0] 3890 IF INKEY(1)<>-1 AND x<620 THEN x=x+1
[68] 3900 IF INKEY(0)<>-1 AND y<380 THEN y=y+1
[EB] 3910 IF INKEY(2)<>-1 AND y>30 THEN y=y-1
[A8] 3920 IF INKEY(9)<>-1 THEN GOSUB 3850:ret=1:RETURN
[F0] 3930 IF x=ox AND y=oy THEN GOTO 3880 ELSE RETURN
[F1] 3940 'PATTERN FILL Loader by Bob Taylor
[BF] 3950 RESTORE 4050
```

```
[2E] 3960 FOR lin=0 TO &210/8-1:total=0:FOR n=0 TO 7:
         READ a$
[31] 3970 byte=VAL("&"+a$):POKE &9040+lin*8+n,byte
[4B] 3980 total=total+byte:NEXT n
[F6] 3990 READ a$:IF VAL("&"+a$)<>total THEN PRINT:PRI
         NT"Error in line"lin*10+4050:PRINT:END
[C4] 4000 NEXT lin
[2D] 4010 speckle$="1001"
[66] 4020 stripe$="10"
[33] 4030 whole$="1"
[6A] 4040 RETURN
[59] 4050 DATA D6,03,C0,32,3D,92,32,41,30D
[24] 4060 DATA 92,DD,7E,04,32,3B,92,DD,3CD
[A1] 4070 DATA 7E,02,32,3F,92,2A,71,B0,2CE
[D4] 4080 DATA ED,5B,6C,AE,3A,06,00,FE,3A0
[72] 4090 DATA 91,28,07,2A,BD,B0,ED,5B,36F
[1A] 4100 DATA 89,AE,25,22,43,92,22,47,2BC
[C8] 4110 DATA 92,EB,22,45,92,DD,6E,00,3C1
[D1] 4120 DATA DD,66,01,7E,B7,CB,47,23,3AB
[CD] 4130 DATA 7E,23,66,6F,7E,FE,3A,38,364
[39] 4140 DATA 04,E6,5F,D6,07,D6,30,CD,3F9
[FD] 4150 DATA 2C,BC,12,13,23,10,ED,CD,2FA
[EB] 4160 DATA C6,BB,CD,F0,BB,CD,2C,BC,5AE
[BB] 4170 DATA 32,34,92,CD,C6,BB,CD,11,424
[C9] 4180 DATA BC,28,06,30,08,CB,3A,CB,2F2
[EC] 4190 DATA 1B,CB,3A,CB,1B,CB,3C,CB,3D8
[72] 4200 DATA 1D,CD,1D,BC,AF,32,35,92,36B
[15] 4210 DATA 32,3B,92,DD,21,3B,92,CB,392
[D6] 4220 DATA 79,28,06,CD,09,92,28,0D,244
[86] 4230 DATA 2B,CB,01,CD,EC,91,3A,34,3AF
[9A] 4240 DATA 92,AE,A1,28,EA,E5,CD,29,4CE
[D0] 4250 DATA BC,22,36,92,E1,E5,CD,26,45F
[2A] 4260 DATA BC,22,39,92,E1,DD,21,3B,3C3
[74] 4270 DATA 92,3E,01,CD,EE,91,CB,09,3F1
[1E] 4280 DATA 30,17,23,CD,09,92,CA,BD,359
[DB] 4290 DATA 91,E5,2A,36,92,23,22,36,2E3
[53] 4300 DATA 92,2A,39,92,23,22,39,92,297
[C3] 4310 DATA E1,3A,34,92,AE,A1,C2,BD,4AF
[45] 4320 DATA 91,CD,09,BB,FE,FC,C8,CB,5AF
[55] 4330 DATA 74,CA,BD,91,E5,2A,43,92,470
[47] 4340 DATA ED,5B,3B,92,3A,41,92,47,369
[7D] 4350 DATA B7,28,03,19,10,FD,ED,5B,350
[62] 4360 DATA 3D,92,19,79,A6,47,79,2F,2F6
[FC] 4370 DATA E1,A6,B0,77,E5,3A,41,92,4A0
[0A] 4380 DATA F5,DD,21,3F,92,CD,EC,91,50E
[AB] 4390 DATA DD,21,35,92,2A,36,92,CD,384
[DB] 4400 DATA 7A,91,F1,F5,32,41,92,3E,434
[BB] 4410 DATA 01,DD,21,3F,92,CD,EE,91,41C
[86] 4420 DATA DD,21,38,92,2A,39,92,CD,38A
[23] 4430 DATA 7A,91,F1,32,41,92,E1,C3,4A5
[5E] 4440 DATA ED,90,3A,34,92,AE,A1,28,3F4
[95] 4450 DATA 05,DD,CB,00,86,C9,DD,CB,4A4
[B6] 4460 DATA 00,46,DD,CB,00,C6,C0,DD,451
[E6] 4470 DATA 2A,47,92,11,FB,FF,DD,19,404
[14] 4480 DATA 3A,45,92,DD,95,3A,46,92,395
[00] 4490 DATA DD,9C,D0,DD,22,47,92,DD,4FE
[9E] 4500 DATA 75,00,DD,74,01,DD,71,02,317
[49] 4510 DATA 3A,41,92,DD,77,03,3A,3D,2DB
[A9] 4520 DATA 92,DD,77,04,C9,2A,47,92,3B6
[41] 4530 DATA E5,DD,E1,ED,5B,43,92,AF,56F
[BF] 4540 DATA ED,52,CB,DD,6E,00,DD,66,495
[A1] 4550 DATA 01,DD,4E,02,DD,7E,03,32,2BE
[3E] 4560 DATA 41,92,DD,7E,04,32,3D,92,333
[0B] 4570 DATA 11,05,00,DD,19,DD,22,47,252
[4B] 4580 DATA 92,C3,BC,90,3E,FF,DD,86,541
[E8] 4590 DATA 02,CB,7F,2B,08,DD,7E,00,2D7
[45] 4600 DATA 3D,DD,77,02,C9,DD,77,02,3B2
[B8] 4610 DATA DD,96,00,D8,DD,36,02,00,360
[B9] 4620 DATA C9,7D,E6,0F,C0,7C,E6,07,464
[92] 4630 DATA D6,05,30,02,C6,05,85,E5,342
[B7] 4640 DATA C5,21,24,92,01,10,00,ED,29A
[72] 4650 DATA B1,C1,E1,C9,00,50,A0,F0,4FC
[D6] 4660 DATA 41,91,E1,32,B2,D2,23,73,3CF
[B2] 4670 DATA C3,14,64,B4,00,00,00,00,1EF
[7D] 4680 DATA 00,00,00,00,00,00,00,00,000
[7F] 4690 DATA 00,00,00,00,00,00,00,00,000
[B1] 4700 DATA 00,00,00,00,00,00,00,00,000
[3F] 4710 MOVE 530,318:DRAW 530,315:DRAW 570,315:DRAW
         570,380:PLOT 530,315
[EF] 4720 WINDOW #2,55,71,2,5:CLS #2:PAPER #1,0:PEN #1
         ,1:LOCATE #1,55,1:PRINT #1," SHADING "
[C5] 4730 PAPER #1,1:PEN #1,0:PRINT #2,"Dense speckle"
         :PRINT #2,"Flat stripes"
[AC] 4740 PRINT #2,"Complete fill":PRINT #2,"EXIT MENU"
[03] 4750 lx=552:ly=336:GOSUB 1740
[91] 4760 IF my>381 AND my<399 THEN shad$=speckle$:high
         =2:wid=2
[1B] 4770 IF my>365 AND my<382 THEN shad$=stripe$:high=
         2:wid=1
[C7] 4780 IF my>349 AND my<366 THEN shad$=whole$:high=1
         :wid=1
[87] 4790 RETURN
```

# Linechecker

## A PROGRAM TYPING AID

All programs in Print-Out have Linecheck codes which are enclosed in brackets at the start of a line. Don't enter them in as they're designed to be used with Linechecker to eliminate errors when typing in programs which appear in this magazine. Please note, all programs will run whether Linechecker is being used or not. For information on how to use Linechecker, please see Issue Three.



# FIRESTONE

This is the first of two text only adventure games produced by Simon Avery. They're all written using the Quill (it's interesting to see that this is still a favourite among amateur adventure writers, now that Adlan has been released), and as most adventure games written using the same 'Creator' are similar, I was particularly interested to see what this game was like.

There was no title screen as such to this game although the screen showing all the information and scenario was quite nicely done - the different style of lettering easily capturing the attention. As this was a text only adventure the lack of graphics was a 'minus' point but I found that the text was written in a rather interesting manner (!) and contained some 'humour' which is good to see (and also appreciated) in this type of game.

Having said that, it was annoying to find that a large number of the major words that were used in the descriptions, weren't understood in the commands.In fact, as a whole, the game appeared to have a limited vocabulary. This led to a number of tricky 'word juggling' exercises in order to find a sentence that was understood by the computer; this is the most frustrating part of any adventure, when you know what to do but are unable to find the correct syntax. Although, a common problem with homebrew games (due often to lack of memory), it is easy to overcome by providing an instruction sheet, complete with a list of words that could be used.

To sum up, the game is a good effort with the Quill and although I've seen and reviewed better text-only games, this did keep my interest & there appeared to be plenty of locations. Firestone is available on tape or disc and the price is a mere 50p is you send a blank tape or disc plus an S.A.E.

# JASON AND THE ARGONAUTS

Although the scenario was not an original one, I thought it was used well and made a change to the usual 'your ship has crashed on an alien planet theme'. For those of you who don't know the legend of Jason & the Argonauts, your quest is to try and recover the Golden Fleece.

The first thing that struck me was that this game had a better vocabulary and set of instructions than Firestone. Again, an imaginative use of language & a number of witty (?) jokes helped to liven up the atmosphere. There was also a large number of locations which were interesting and several included character interaction (always an excellent feature in any game). All of the puzzles were logical but some were quite devious. There was enough to keep you busy for many hours, I'm sure. One suggestion would be to provide free hint sheets on request for the entire game as this means that the player can see the complete game and it relieves the frustration that some of the problems can cause.

Jason and the Argonauts also costs just 50p plus a blank tape or disc and a S.A.E. To see the other games from Simon Avery turn to the SMALL ADS section. Both JASON AND THE ARGONAUTS and FIRESTONE are PUBLIC DOMAIN but you can obtain them from Simon Avery at MARDEN FARM, OLD EXETER ROAD, CHUDLEIGH, SOUTH DEVON.

**CRIMSON —** This is the first of three small, but nevertheless good, games from Adrian Sill who has only recently tried writing games. I have to say that while I like adventure games, it is good to review a homebrew game that is different every now again.

Crimson is an arcade game in which the aim is to steer a miniature spacecraft through 19 screens. Various obstacles lie in the way and block your clear passage from one end of the screen to the other.

The title screen was quite good but I found that some of the colours were hard to see on a green screen. Although the game is a simple one, the instructions were clear, neatly printed and easy to understand.

The screens themselves were well done but not too complicated and so they varied in difficulty from the incredibly easy to some that were very hard. The game did lack some music and the so called spacecraft was in fact a dot on the screen. An improvement would be to make the spaceship bigger & therefore, make the gaps between the obstacles larger.

The game was very addictive for the first few times but I rather fear that it would become tedious after longer. Overall it is a well produced game.

**TRAITOR —** The second of the Sill trio. In this game, you've been a traitor to your country & have been given the chance to live by going through four of the President's mazes. In order to finish the screen, it's necessary to collect fifteen gold bars that are littered through the maze & take them to the exit. When this has been done the maze is completed and it's onto the next one.

The scenario could hardly be called original but it was the quality of the game that impressed me. The graphics were good & quite clear although I thought that the maze & map could have been made bigger so they filled the whole screen. Also, some of the graphics were slightly faint so could not be seen unless in a dim room. That said, it was an excellent game which had me hooked for some time.

**SNOOKERED** The final one in Adrian Sill's collection is very different from the others as it is a strategy game based on snooker. In it you are competing for different trophies and trying to improve your world rating at the same time. You are given a number of options, such as save game, view progress, see status and start the cup. When a tournament is coming up, you are allowed a certain number of weeks in which to practice your potting, defensive and safety play, or alternatively, to rest. Depending on the number of weeks you practice, your ratings in the various skills will change, thus giving you a better chance at success. Once the tournament starts, you watch your progress through the cup (more often than not, you go out in the first round to begin with) and then you start all over again. Although the game is very nicely presented, it lacks any real depth and there's no incentive to play again after a few games. It's still worth buying just to see the beautiful bold text at the beginning. The price is just £4.99 for all three of Adrian Sill's games and are available on disc only, from 19 Sherwood Drive, 5 Lane Ends, Skellow, Doncaster, South Yorks., DN6 8N7.

## CASINO BLACKJACK

Finally for this issue, we come to two programs written by Barrie Snell. One is an incredibly detailed and also realistic simulation of Casino Blackjack and the other is a wordsearch utility. The first thing to note about Casino Blackjack is that doesn't work on a 464 as it utilises some of the 6128's extra commands and is available only on disc, at a price of £4.50 including a disc or £2.50 if you send a disc as well. The cost is for both programs (Blackjack and Wordsearch) on the same disc.

After loading, you are faced with a menu from which you can choose to read about the program, see which variables are used, view the rules for play & even the world-wide variations on Blackjack or play the game. The first four options are very detailed and give a potted history on Casino Blackjack, throughout the world, & should prove invaluable to anyone who is very interested in Blackjack. Once you have decided to play the actual game, you are allowed to make a number of decisions about exactly what rules the game will be played under.

First, you have to say how many people are going to play & whether you want the computer to also play a hand (it automatically plays the dealer). Next, you have to decide on the amount of money you wish to start off with (upto £2000!). Then, it is necessary to decide which laws you intend to use; either British or American rules or even your own local rules. You are then almost ready to start but first you must choose if you want to use a random shuffle or one of the pre-set patterns stored on the disc (part of the skill in Blackjack is remembering which cards have gone and which are left). Finally, you can start playing.

The computer plays an intelligent hand & the whole playing routine is very well thought out and designed. All your options on a particular hand are shown, and they can be selected at the touch of a key; this is useful as there are are a lot of special features (such as doubling,surrendering,splitting,etc) as well as the normal Stand or Hit. The cards themselves are very well drawn & it's put together extremely well.

For the novice all of the choices in the preliminary stages may be a touch confusing but there is often an easy option which passes you onto the next part There is even a demonstration mode where the computer plays all of the hands !! When you finally decide to end your session at the Blackjack table you're given a detailed statistical analysis of your play including your win/lose rate. It's a very polished and complete program. Excellent for anyone who likes a game of cards and a spot of gambling !!!

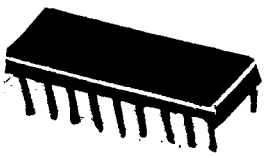
## WORDSEARCH

This program allows you to solve those fiendishly tricky wordsearch puzzles that often appear in competitions. There is not a lot one can say about this program as it's helpful, relatively fast, full of useful functions, well developed, neatly presented and does what it sets out to do and this in, my opinion, is what counts. Whether it is worth buying, really depends on how many wordsearches you are planning to solve. Still, along with the truly brilliant Blackjack it's excellent value for money. Both programs are available from Barrie Snell, 19 Rochester Road, Southsea, Portsmouth, Hants PO4 9BA.

The cost is £4.50 including disc and this price is for both programs.

Machine ....Code....

# Programming your Z80 GRAPHICS



This issue, we are going to look at writing a graphics routine. To do this, we'll need to look at several new firmware commands. But first, we have to look at how the CPC handles graphics.

In BASIC there are two ways of specifying a point on the screen, either by using relative or absolute coordinates. In Machine Code, there are two firmware calls for the two types of graphic movement, but before we look at any specific examples, it's advantageous to have a brief overview of the CPC's graphics.

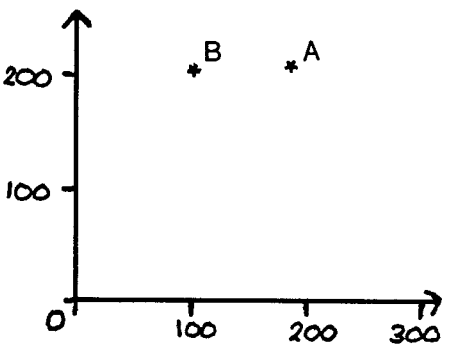
In all three modes the screen is 640 points across and 400 points high and each of the points can be addressed individually by the user (this isn't to say that each point can be plotted individually on the screen). However, as this is the case the coordinates have to be stored in register pairs (as 16 bit numbers) because 640 and 400 are too large to be stored in a single register. Therefore, the horizontal (the X) coordinate is stored in DE & the vertical coordinate (Y) in HL.

These coordinates are known as USER coordinates and the computer then converts them into BASE coordinates before actually plotting onto the screen. However, fortunately for us, we do not have to worry about this conversion, unless we want to access the screen in a more direct manner.But for the time being, we will be sticking to leaving all the tricky calculations upto the computer.

First of all we will look at some simple applications based on the Machine Code equivalents of the MOVE and DRAW commands. The addresses to call for these two commands are &BBC0 and &BBF6 respectively. Thus the short Machine Code program below draws a box at the centre of the screen using absolute coordinates.

```
                ORG &8000
11 0E 01        LD DE,270       ; let the x coordinate be 270
21 96 00        LD HL,150       ; let the y coordinate be 150
CD C0 BB        CALL &BBC0      ; then move to the place denoted by DE and HL
11 0E 01        LD DE,270       ; let the x coordinate be 270
21 FA 00        LD HL,250       ; let the y coordinate be 250
CD F6 BB        CALL &BBF6      ; then draw to the point denoted by DE and HL
11 72 01        LD DE,370       ; let the x coordinate be 370
21 FA 00        LD HL,250       ; let the y coordinate be 250
CD F6 BB        CALL &BBF6      ; then draw to the point denoted by DE and HL
11 72 01        LD DE,370       ; let the x coordinate be 370
21 96 00        LD HL,150       ; let the y coordinate be 150
CD F6 BB        CALL &BBF6      ; then draw to the point denoted by DE and HL
11 0E 01        LD DE,270       ; let the x coordinate be 270
21 96 00        LD HL,150       ; let the y coordinate be 150
CD F6 BB        CALL &BBF6      ; then draw to the point denoted by DE and HL
C9              RET             ; return from Machine Code
```

We will now look at the relative versions of these commands. The addresses are &BBC3 for GRA_MOVE_RELATIVE and &BBF9 for GRA_LINE_RELATIVE. A slight problem, however, rears its head when you are trying to use these commands. To show this difficulty clearly, look at the small diagram below.




Using absolute coordinates, point A is located at 200,200 and point B at 100,200. However if we use relative coordinates, B is at -100,0 relative to point A. In other words, to draw a line from A to B would require the commands: MOVER 200,200:DRAWR -100,0 (if we were using the relative system). In BASIC, to signify a negative number all you do is put a minus sign in front, but in Machine Code, a 16 bit register can only hold values of between 0 and 65535 (ie no negative numbers) Fortunately, some clever people came up with the concept of signed numbers and 2's complement. Now, if you're lucky and have a decent assembler, you can just enter negative numbers with no trouble. But in next issue's machine code, we will be explaining all about the wonders of Two's complement and signed numbers. However, 2's complement will explain some of the funny numbers in the BASIC poker). But still here's the 'relative' version that will draw a box on the screen:

```
                ORG &8000
11 0E 01        LD DE,270       ; let the x offset be 270
21 96 00        LD HL,150       ; let the y offset be 150
CD C3 BB        CALL &BBC3      ; move relative to the current cursor position
11 00 00        LD DE,0         ; let the x offset be 0
21 64 00        LD HL,100       ; let the y offset be 100
CD F9 BB        CALL &BBF9      ; draw relative to the current cursor position
11 64 00        LD DE,100       ; let the x offset be 100
21 00 00        LD HL,0         ; let the y offset be 0
CD F9 BB        CALL &BBF9      ; draw relative to the current cursor position
11 00 00        LD DE,0         ; let the x offset be 0
21 9C FF        LD HL,-100      ; let the y offset be -100
CD F9 BB        CALL &BBF9      ; draw relative to the current cursor position
11 9C FF        LD DE,-100      ; let the x offset be -100
21 00 00        LD HL,0         ; let the y offset be 0
CD F9 BB        CALL &BBF9      ; draw relative to the current cursor position
C9              RET             ; return from Machine Code
```

As you can see, a program to draw a complex shape on the screen would soon become very lengthy indeed if we kept using MOVEs and DRAWs so printed below is a routine which you can include in your Machine Code toolbox (this a collection of useful routines, which can be easily incorporated into a larger program with a minimum of trouble at a later date). All the information for the coordinates, is stored in the data at the end, in the following order :

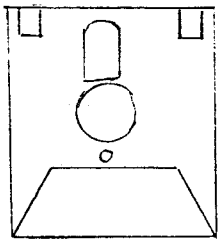
X coordinate, Y coordinate, code ... and then it repeats.

If the code is 1, it means that the computer is to move to the specified coordinates but, if the code is a 2, the computer will draw instead. But if the code is &FF then this signals the end of the data. Note that each of the entries has two bytes (making it a sixteen bit number) including the code and numbers which are less than 255 (even though they don't need to be). This just makes the program less complicated but if you use an assembler it shouldn't bother you. Just enter the data without worrying about two bytes or otherwise. However, the only bit of padding you need to insert are the two zero bytes just before the end of data marker; this is because no coordinates are needed.

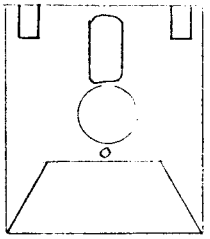
```
                ORG &8000
01 2B 80        LD BC,data
0A        .loop LD A,(BC)
5F              LD E,A
03              INC BC
0A              LD A,(BC)
57              LD D,A
03              INC BC
0A              LD A,(BC)
6F              LD L,A
03              INC BC
0A              LD A,(BC)
67              LD H,A
03              INC BC
0A              LD A,(BC)
FE FF           CP &FF
C8              RET Z
05              PUSH BC
FE 01           CP &1
CA 22 80        JP Z,move
CD F6 BB  .draw CALL &BBF6
C1              POP BC
03              INC BC
03              INC BC
C3 08 80        JP loop
CD C0 BB  .move CALL &BBC0
C1              POP BC
03              INC BC
03              INC BC
C3 03 80        JP loop
0E 01 96  .data DW 270,150,1,270,250,2,370,250,2,370,150,2,270,150,2,0,0,&FF
00 01 00
0E 01 FA 00 02 00 72
01 FA 00 02 00 72 01
96 00 02 00 0E 01 96
00 02 00 00 00 00 FF 00
```

Most of that should be fairly self-explanatory as it uses no new programming techniques. However, one thing to note is the way in which it puts a 16-bit number into a register pair. The reason for an unusual method, is that there is no way of doing it directly eg. LD HL,(BC). It is therefore necessary to do it in a number of stages. First A is loaded with the low byte of the number & this is then transferred to the L part of HL. This is then repeated, except that A holds the high byte of the number, & this is then moved into the H part HL. Thus loading HL with a 16-bit number stored in memory by using 2 simple operations. This month's challenge is as follows;given that the address for GRA_PLOT _ABSOLUTE is &BBEA, and that it needs the same information as for DRAW and MOVE ABSOLUTE, try and add a plotting routine, or even convert it to use relative coordinates (GRA_PLOT_RELATIVE is at address &BBED) or possibly do both! Just have fun experimenting with Machine Code until next issue! Remember in our birthday issue, we hope to have our own Machine Code Assembler.If it turns out to be too large and complex we'll try and include it on our program tape/disc.

The numbers in the lefthand column are for use with the BASIC poker (on every program tape or disc) and should not be entered into an assembler. Similarly, the text shouldn't be entered into the BASIC poker.



# Getting to grips with CPM



**by Richard Sergeant**

Unfortunately, due to other outside commitments Richard Sergeant is unable to continue his excellent series on CP/M Plus in the future, but has taken time to answer some of the queries that were raised by his article in Issue Five. So here are his answers, along with some of my own thoughts & comments about CPM+.

1) A question that was often voiced was 'What about CPM 2.2? Why not cover that as well?' In my opinion, CPM 2.2 is very limited and inferior to CPM Plus. However, most programs that were written for CPM 2.2 will run successfully under CPM+, but not the other way round. I feel that the little extra effort required to allow CPM+ to run on a 464 or 664 is fully justified by the extra facilities available to the user.

It may interest those of you who have recently upgraded from a 464 to a 6128 by the addition of a 6128 ROM Chip, that CPM Plus can be purchased from W.A.V.E. at £17.90 including postage (Tel: 0229 870000). Another offer for present owners of CPM Plus is to have the CPM+ software transferred onto ROM. Many people have found this more convenient than having to boot their Master Disc (copy) as the software is available instantly & doesn't take up any disc space. On top of this the double ROM pack includes plenty of extra commands. If you want more information the people to contact are Graduate Software, 14 Forrester Avenue, Weston-on-Trent, Derby DE7 2HX (Tel: 0332 702993) and the price is £24.95. Please note that they need you to send your CPM+ Master Disc as proof that it is not breaking Digital Research's copyright.

---

2) Another common question was Why did I include examples of non-CPC machines ? The reason behind this was to demonstrate that many CPM Plus programs will port across to different machines provided, of course, they can boot this disc operating system (DOS). For example, a CPM PD file written on a Spectrum +3,Osborne or PCW9512 will happily work on a CPC - the only thing that might need altering are screen handling codes & there is often a customiser program that will alter these. The only other difficulty is reading non-CPC discs and this can be overcome easily - there is plenty of PD software available to allow PCW discs to be read by a CPC, and all PCW's can happily read CPC discs.

---

3) Another comment which follows on from the last point raised was 'Why not review some CPM programs?' If you want reviews of CPM programs, the last place to look is in a CPC magazine. Amstrad Action and Amstrad Computer User are a waste of time if anyone wishes to read about CPM PD or any serious commercial programs and utilities. If you want CPM program reviews buy some of the many back issues of PCW glossy magazines. A large number of the CPM programs for the PCW 8256 or 8512/9512 will also work on a CPC 6128 (under CPM+). For example, Arnor produce Protext CPM+, Maxam II, C and BCPL which will run on all the above computers.

---

4) A few readers asked me to do something for the beginner. However, for anyone who is new to CPM+, I give this piece of advice: I have found that the only way to learn about computers is to use and experiment with the damn things day in & day out for years. Probably, one of the major reasons why CPM is shunned by so many people, especially newcomers to computing, is because it's so different to BASIC & does not have understandable error messages to explain what the problem was. However, like all things to do with computers, CPM can be mastered (in the end!!!). There are several excellent books on CPM around and, remember, they do not have to be CPC specific. Sorry, but there is no easy option, but listening to people like me and picking up hints and tips should help a bit.

---

5) Talking of hints and tips I received a letter asking how you get CPM Plus to recognise other USER areas. The answer to this is as follows.

At the A> prompt type USER A10: (you don't need the preceding ¦ symbol as in AMSDOS; also note that the colon is most important). The computer responds with 10A> and you are now in drive A, user area 10. To return to user area 0, type:

USER A10: (Note. if you have only one drive, the A can be omitted)

If a *.COM file, such as DIR.COM, is SET to SYSTEM and stored in USER 0 then it can be called and will run in any other user area (ie USER 0 to 15) If you type DIR in user 0, any file SET to SYSTEM will not be shown; you must type DIRS at the prompt, and then all the SYSTEM files will be shown & non-SYSTEM files will be invisible.

If you ever use Locomotive's CPM Plus 'MALLARD BASIC' then a command, such as OPTIONS FILES"10B", needs to be used to switch to the B drive and user area 10. In conclusion should you consider a 3.5 inch or 5.25 inch high density 2nd disc drive running RAMDOS then file maintenance, by using different user areas, is almost essential.

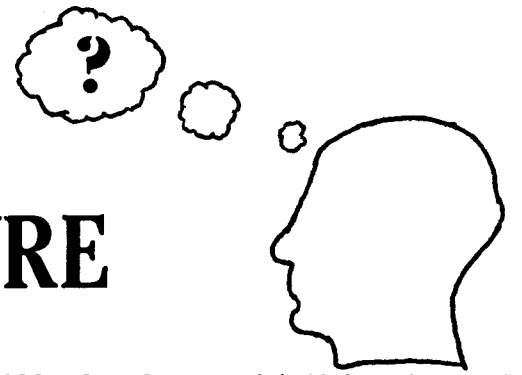
---

I hope that CPM+ seems a little more manageable now, & that some of you have been tempted to use your CP/M+ disc for more than formatting and copying discs. Remember with time and experimentation, all of CPM's secrets can be grasped !!!

If you do want this series on CPM to continue all you have to do is to write in and tell us - we will do our best to try and find a replacement. And, if you know about CP/M, then why not write and tell us; who knows you might find yourself writing the next CPM column.

PRINT-OUT

# LOOKING to the FUTURE



As many of you will be aware, next issue will also be our birthday issue & we hope it will be something rather special. As well as all of the regular articles there will be competitions, special offers, questionnaires and with a bit of luck, we will be printing our very own Machine Code Assembler for anyone who wishes to learn assembly language without spending large sums of money.

Now that the first year of the magazine is coming to an end, we thought it would be an interesting idea to reflect on all that has happened in the past 12 months and, more importantly, to look forward to the next year.

---

Print-Out was launched in the September 1989 issue of Amstrad Action & was reviewed in their fanzine section, 'The Write Stuff'. The first issue was produced on a 1950's manual typewriter but, because of the difficulty in correcting typing errors, we hastily produced Issue Two on an electric typewriter. However this still proved unsatisfactory and Issue Three was written on a CPC 464 using Protext (on ROM) and was printed on a Star LC-10 colour printer. The only drawback with this was that all the documents had to be saved to tape and this made it very time consuming. So, for Issue Four, a disc drive was added to the 464 & a 6128 ROM was hung on the back (giving both a CPC 6128 and 464 at the flick of a switch).

This was such an improvement, that both Issues Five & Six were produced in the same way. The only real advance in Issue Five, was the inclusion of screen-shots which were captured using a Multiface Two & then printed using a graphics dump and a black and white ribbon.

The next step forward will be the use of Qualitas Plus to improve the print quality of Print-Out. It will not only allow a far greater number of type-styles but will also enable us to print using 'micro-justification', which will help in producing columns for certain articles.

---

Enough of the past, now to the future! During the next year, we hope to be able to set up a network of CPC users, who will be able to help one another and pass on their ideas through the magazine. This will enable us to solve readers' problems far more quickly and efficiently than at present and will also provide contacts for people to get in touch with in their area. Unfortunately this will not happen for several months yet because of the large amount of administration required. Further details will be printed in a later issue of Print-Out.

We are also hoping to start an advertising campaign in the national magazines, such as Amstrad Action. Already we have been given a large review in the CPC specific column of New Computer Express that was written by Rod Lawton, the editor of Amstrad Action.

Many new ideas are being discussed at present, and the next year will hold many surprises for Amstrad users - not least the impact of the new CPC Pluses.

---

ADVANCED MODE

# *An intro to* RSXs *(Part 2)*

## by Bob Taylor

Last time, we looked at initialising Resident System Extensions and at how to handle any Errors which may arise. We will now go on to deal with data input and output for RSXs.

Although some RSXs require no external data to work on, such as |CAPSON & |CAPSOFF in the last article, most do & it is a good idea to be acquainted with the full extent of what's possible.Certain elaborate methods of passing strings with 464 have been simplified for the 6128, so any differences will be detailed as we go along.

Data is 'passed' to the RSX in the form of two byte values which represent the Parameters, which are typed after the RSX name (and are separated from each other and from the name by commas).

Fundamentally, an RSX is a command and as such, its name cannot be used in the usual manner of functions. That is it cannot be used to qualify BASIC commands and statements directly in the way for instance that PEEK does in :

```
PRINT PEEK(&AC00)
or a%=PEEK(&AC00)
or POKE &8000,PEEK(&AC00).
```

However, there is a facility present with RSXs which would allow the value of PEEK(&AC00) to be assigned to a variable which can then be the object of the above examples. This facility is the use of the operator @, in conjunction with number or string variable parameters and these options are shown below. So then to the types of data which can be passed. They are:

1) NUMERIC EXPRESSION - If the result of this is greater than -32768.5 and less than 65535.5, the result is rounded and passed as a whole number. If outside this range an 'Overflow' error results. Also, as with only a two byte result, values from -32768 to -1 (ie. signed integers) are expressed in exactly the same way as those from 32768 to 65535 (unsigned integers), a decision to use signed or unsigned must be taken before writing the RSX routine.

2) NUMERIC VARIABLE / NUMERIC ARRAY ELEMENT - Whether the Variable is a Real or an Integer one, its value is treated like the numeric expression above.

3) STRING EXPRESSION - On the 464, the resultant string must first be assigned to a String Variable which then needs the @ operator when used as a parameter (eg a$="ABC"+b$:|RSX,@a$). With this assignment, the string variable used then operates as explained below in the section on string variables. However on a 6128, the expression can be directly present as the parameter (eg |RSX, "ABC"+b$). For compatibility, it's also permissible to use the form required by the 464 as above.

In either case, the parameter is passed as an address, at which will be found a three byte block of data called a String Descriptor.The first byte of this will give the length of the resultant string while the other two bytes are the address of where in memory the result string itself is held.Note that the two addresses just mentioned are not the same thing. The contents of the string can be ascertained by the RSX using the information present in the Descriptor.

4) STRING VARIABLE or STRING ARRAY ELEMENT - For the 464, this must be preceded by the @ operator. For the 6128 the @ is unnecessary, but again may be included for compatibility with the 464. For both, the data passed is again the address of the location of a String Descriptor. From this, the string's contents can be obtained for the RSX.

   Unlike the result of the 6128's string expression (which exists only temporarily for the duration of the RSX routine), the Variable or Array Element is finite, and so it becomes possible to modify either the string itself or the Descriptor in some way in order to pass a result out of the RSX. Thus an RSX, whilst still a Command can be used in a Function role. The ways in which the result of the RSX can be passed out are:

   a) Any characters in the original string can be altered.
   b) The length byte in the Descriptor can be reduced, resulting in a shortened string. If the address part of the Descriptor is left unchanged, this new string will consist of the first part of the original; if the address is moved to point to another character of the original string the reduced string will start at this character.

   ---

   The length byte mustn't be increased nor must the address be moved such that the string would extend beyond its original characters (the reason for this is that in the areas where strings are stored,bytes are present which aren't characters). It follows from this, that the original string should be long enough to hold any result required.

   ---

   c) A new string can be set up elsewhere in RAM, & its length and start address installed in the Descriptor. The only limitation is the usual one of a maximum length of 255 characters. Evidently, such a string must be protected from over-writing or erasure while in use.

5) @NUMERIC VARIABLE/@NUMERIC ARRAY ELEMENT - As with string variables or array elements, the @ operator gives an address as the data. But here, the address points directly to the value of the variable/element in memory and not to an intermediate Descriptor.Since the variable or element could be of either the Integer or Real type, the value pointed to will be in two or five byte form, respectively. This allows floating point values to be handled by the RSX in addition to Integer ones as well as providing the ability to pass out either type of value.

Of course, the type of variable/element input as a parameter must match the type that the RSX routine was written for. With a variable, there is a type byte immediately before the value: &01 for Integers or &04 for Reals. It is possible to use this byte in order to differentiate between the two types & take alternative action.With array elements, the type byte is within a data block which precedes the various elements & so isn't available for checking.

6) @STRING VARIABLE OR @STRING ARRAY ELEMENT - As (4) above

In order to use @ in a parameter, it's necessary first to have set up the variable or element by assigning a value or finite string to it; failure to do so results in an 'Improper Argument' error message.

## PARAMETERS IN THE RSX

On entry to the RSX routine from BASIC, the two byte parameters will have been stored on the Machine Stack with the first, highest up, and the others in sequence downwards. The IX register will be set to point to the last parameter which will be the lowest on the stack. As two byte values are always stored in memory with the low byte first, IX will point to the low byte of the last parameter.Assuming three parameters are passed, the following indexing will apply:

(IX+5) gives high byte of first parameter
(IX+4) gives low byte of first parameter
(IX+3) gives high byte of second parameter
(IX+2) gives low byte of second parameter
(IX+1) gives high byte of last parameter
(IX+0) gives low byte of last parameter

If there's a different number of parameters to be passed, the upper limit of displacement of IX will need to be adjusted accordingly. Other bytes on the stack outside this range, are not from parameters and should not be accessed.

Parameter entries on the stack cannot be obtained by POPping (as the Stack Pointer is usually six locations below the last parameter entry); therefore the only method of access is to read the bytes from where they are and this is best done using the IX register with displacements.

If you have a fixed number of parameters for an RSX, the indexing is fixed and entries can be read from the parameter data block in any order you like; it is not usual to move the base address of IX, and so a certain displacement will always point to a specific parameter's byte.

On the other hand, some RSXs allow a variable number of parameters (in the same way that BASIC's 'SOUND' command allows between two and seven parameters). Under these circumstances it might become essential to move the IX base address in order to access the correct parameter and it is usual to do so with

```
INC IX:INC IX
```

which will move two locations up the data block to the next one. But moving the base need only be done for each parameter which is optional, and present - when it's omitted on another occasion, the base shouldn't be moved. Those parameters which must always be present can be accessed using fixed displacements as above.

Once used, the bytes previously occupied with parameter data are redundant and can be re-used to temporarily store other data inside the RSX routine - the displacements of IX will be useful again. It goes without saying that any other bytes on the stack, not previously used for input data, must not be altered in any way!

One useful but undocumented fact which we can make great use of is that on entry to the RSX routine from BASIC,the DE register pair will hold the two byte value passed as the last parameter.Therefore, it will often not be necessary to access (IX+1) and (IX+0). This only applies when there are parameters; if there are none, then DE will hold the address of entry which we will find useful when we come to re-location in the next article.

---

Also on entry to the RSX, the A register holds the number of parameters present with the bar command. Since, some RSXs must have a certain number of parameters to work on, this register will need to be tested. With those RSXs which allow a variable number of parameters, it's simplest to allow those not always required to be missing from the end of the list; then the sequence of parameter usage is easier to remember than if they are omitted from the start of the list.In these cases too, testing the A register will show how many,and therefore which, parameters are left off.

In theory, parameters could be missing from wherever we choose, but it is easier to comprehend & check, if there's a logical sequence to their omission. It is impossible to detect a situation where the same number of parameters may be omitted from different parts of the parameter list, since the only means of detection is via the value held by A which must not be the same for two different combinations of parameters. All of this means that with a given value in A a known sequence of parameters must be present.

---

Incidentally, there's no built-in check made to ascertain whether the parameter entered is of the correct sort (eg string as opposed to number). The type byte, preceding a variable's value (that was mentioned earlier) extends to all string variables which have a byte of &02 before the string descriptor;

This can be used with parameters which are variables, but not with expressions or array elements. Apart from these, there is an indirect method which can be used to prove that an entry is not a string, but that is all:

The string descriptor first byte holds the length of the string. The last two bytes of the descriptor point to the start of the string proper. But, two bytes before this start position, the string's length is again stored. If a parameter is expected to be a descriptor address, then these two lengths can be compared and if found to be different can be taken for certain that the parameter is not that of a string. However, if they do match, it is not guaranteed that a string is present, as it could be a remote coincidence that the bytes are the same.

There are some other checks that can be made, but still without absolute confirmation of a string:

1) The byte before the string proper (ie after the length byte) should contain &00. It is actually the length high byte, but as strings cannot be longer than 255 characters, it is redundant and must contain &00.
2) A string descriptor cannot be present below &0176 so the address OF the string descriptor cannot have a high byte of &00.
3) Similarly, a string cannot be stored below &0176 so the high byte of the address IN the string descriptor cannot be &00.

Usually no check is made in RSX routines for the type of parameter entered, it being left to the BASIC programmer to enter the correct type!

---

All the preceding information on parameters also applies to routines CALLed from BASIC. Further, it's possible to CALL an RSX routine, both before & after, it has been initialised (and after too); and this might be useful for debugging purposes (of course the address of the routine must be used).

---

The following routine provides two new RSXs comprising a double byte POKE and a similar double byte PEEK and incorporates some of the points raised:

```
        ORG  &8000
.intrsx LD   HL,&8000  ;the address of the start of this initialisation
                       ; routine. .datblk starts at the second byte of this
                       ; instruction.
        LD   (HL),&C9  ;store a RET byte in place of the LD HL instruction
                       ; byte; unlike the previous article, we are no longer
                       ; using an extra NOP byte.
        INC  HL        ;HL now points to .datblk for KL LOG EXT.
        LD   BC,comtbl ;address of the command table.
        JP   &BCD1     ;return directly to BASIC from Firmware Call KL LOG EXT.

.namtbl DEFB "PEEK","2"+&80
        DEFB "POKE","2"+&80
        DEFB &00

.comtbl DEFW namtbl-intrsx
.peekjp JR   peek2
        DEFS 1         ;pad out PEEK2's entry to three bytes.
.poke2  CP   &02       ;start POKE2 here in place of its entry.
        JR   NZ,parerr ;error if not two parameters with the RSX.
        LD   H,(IX+3)  ;get first parameter -
        LD   L,(IX+2)  ; HL=addr to POKE into.
        LD   (HL),E    ;when entering an RSX routine with parameters, DE holds
                       ; the last parameter, ie 2 byte value to POKE.
        INC  HL
        LD   (HL),D    ;POKE 2 byte value into addr and addr+1, low byte first.
        RET            ;to BASIC.
```

## A TWO BYTE POKE AND PEEK ROUTINE

```
.peek2  CP   &02
        JR   NZ,parerr ;must be two parameters with the RSX.
        LD   H,(IX+3)  ;get first parameter -
        LD   L,(IX+2)  ;HL=addr. to PEEK from.
        LD   C,(HL)
        INC  HL
        LD   B,(HL)    ;BC=contents of addr (in C) and addr+1 (in B).
        EX   DE,HL     ;on entry, DE held the last parameter (addr of value in
                       ; integer variable); now HL does.
        LD   (HL),C
        INC  HL
        LD   (HL),B    ;store PEEKed bytes in variable (earlier=low byte).
        RET            ;to BASIC.

.parerr LD   HL,errmes ;HL -> error message 'Check Parameters'.
        CALL prtstr    ;print the error message.
        LD   A,33      :ERR number.
.errcom LD   C,0       ;for ROM 0 (BASIC).
        LD   HL,&CB55  ;(&CA93 for 464); BASIC's error handling routine in ROM.
        JP   &001B     ;to FAR CALL to ROM; the error handling routine alters
                       ; the stack so doesn't REturn.

.prtstr A,(HL)         ;get first/next char. of string to print.
        OR   A
        RET Z          ;if string finished.
        CALL &BB5A     ;print char.
        INC  HL        ;step on to next char.
        JR   prtstr    ;for next character.

.errmes DEFB "Check Parameters",&00
```

The syntax for these RSXs is as follows:

|POKE2 ,<Address to POKE> ,<2 byte value to POKE into Address>

The low byte of the value will be POKEd into Address; the high byte of the value (or 0 if the value is less than 256), will be POKEd into Address+1.

|PEEK2 ,<Address to be PEEKed> ,<@Number Variable or @Array element for result>

The number variable or array element must be an Integer one either by using % or after using DEFINT. It must exist before use here;ie it must have been assigned a value (even 0 will do).

The byte PEEKed from Address will become the low byte of the variable's value, while its high byte will be the byte from Address+1.

---

Next issue we will continue this series on extra commands by detailing methods for re-locating RSX routines.

---

SERIOUS USE

# TEARAWAY HACKING TOOL for a MULTIFACE

One of the most common pieces of computer add-on hardware for the CPC, is the Multiface Two, one of the infamous 'black boxes'. Often, its sole use is to transfer the owner's tape based software onto disc for speed of loading, but it does have a secondary use which many people seem to be exploiting nowadays. I'm talking about finding cheats and pokes for a game.

However, anyone who has attempted to use the Multiface Two's tool box for serious hacking will have realised just how frustrating it is. Without a search mode, a disassembler and a decent memory editor it is like looking for a needle in a haystack. Up until now, there has been one program, the Insider, which can be installed into the Multiface & used to provide the necessary extra features. But now, Tearaway has been written which is designed to do all that the Insider does and more. The question is, does it succeed.

Both programs need one disc drive fitted and a Multiface Two device. This means that neither program is available on tape and this is a serious drawback for many people. In order for the features to be used efficiently and successfully, both programs assume a bit of knowledge of Machine Code although all of the functions are accessed from simple menus.

The first major difference is the way in which the two programs work. With the Insider, a relatively short program is loaded in the Multiface's extra RAM, and then whenever a function is selected, the disc needs to be accessed to load in the corresponding piece of code. The major advantage of this is that none of the computer's memory is corrupted by Insider, although the near constant disc access is both annoying and time consuming. Tearaway, on the other hand, loads into one of the CPC's banks of extra memory (this meaning it will only work on machines with at least 128K of memory). The next difficulty with this, is that you cannot load 128K programs using the conventional manner as this would overwrite Tearaway's code. However, a way round this is listed in the instructions for the program although it's success is not always guaranteed. To access both programs, all you have to do is press the RED Multiface button after they have been installed (it is still possible to use the normal Multiface menu by pressing any key at the same time as pressing the RED button.

Before putting the two programs too work, I glanced through the functions that were available. Insider boasted a Z80 disassembler (which could handle all the documented and undocumented opcodes), a find facility to search for text or opcodes, a register display to show all Z80 registers,etc, the ability to enter pokes once they had been found, memory could be shown as numbers or ASCII & all of this could also be outputted to a printed if required.

Tearaway specification's were very impressive. It matched Insider in every department and also included the ability to view the memory as a graphic image, display the hardware palette, CRTC registers, Rom and Interrupt Status and the Mode setting (all of which could be altered at will), copying memory & a way to insert 'null bytes' (unknowns) into the search routine. The important question now was 'What would they be like to use?'

Once you've loaded the game you wish to dissect, you're faced with a large number of options. In Tearaway these are clearly laid out and well explained in the copious notes supplied on the disc (they can be printed out to form a small leaflet) whereas with Insider they're crammed into a tiny box at the top of the screen. I then put both programs to the test by finding an infinite lives cheat for Netherworld.

Fortunately, Netherworld does not use a complicated system for storing its lives (although, I did not know it at the time). After a few minutes of getting used to the way Tearaway worked, I decided to start by trying to look for some of the commonest ways of storing and decreasing lives (half a dozen methods are given in the Tearaway instructions). In the end it turned out not to be any one of these but a variation. However, I entered the search menu and asked Tearaway to find all occurrences of LD A,5:LD (xxxx),A where xxxx was an unknown value. In well under a second, it had come up with a number of promising looking addresses (see figure 1). I then disassembled each in turn until I found a suitable value with which to replace xxxx (see figure 2) which turned out to be &64F5. It was then a matter of looking up all further references to this address, then finding one that included a DEC or SUB command, replacing the SUB with a couple of zero bytes and then returning to test the game out. It worked first time !!!

The Insider also came up with the same results but after far longer & with more work done by myself. The search routine is noticeably slower, taking over 20 seconds on the same checks as above and unknowns are not allowed. The printing worked but was very unclear and difficult to understand (see figure 4) and even the disassembler (see figure 3) had a few quirks when printing out.

In conclusion both programs worked & achieved the same results eventually but Tearaway wins on the user-friendliness stakes. Tearaway lets you deal with the Z80 in a far more comprehensible manner and is a real boon to any M/C programmer, hacker, or just the plain inquisitive. TEARAWAY costs £10.50 (or £7.50 plus a blank disc) and can be ordered from CPC NETWORK, 3 The Cottons, Outwell, Wisbech, Cambs PE14 8TL. INSIDER costs £14.95 (but also see the special offers that are available) For further details contact ROMANTIC ROBOT on 081-200-8870.

```
1) Tearaway search for 3E,05,32
String : 3E 05 32 00 00 00 00 00 00 00 00 00 00 00 00    >.2.............
Null byte : 00
String Found at locations :-
&17CB :   &21D5 :   &2684 :   &270F :   &290E :   &637D :   &6674 :
```

```
2) Disassembly using Tearaway
2908  LD HL,&662C     ; &21 &2C &66    ;!,f
290B  CALL &6614      ; &CD &14 &66    ;..f
290E  LD A,&05        ; &3E &05        ;>.
2910  LD (&64F5),A    ; &32 &F5 &64    ;2.d
2913  CALL &72E1      ; &CD &E1 &72    ;..r
2916  CALL &2A21      ; &CD &21 &2A    ;.!*
```

```
3) Disassembly with Insider
2908  21 2C 66    LD HL,662C    !,f
290B  CD 14 66    CALL 6614     M.f
290E  3E 05       LD A,05       >.
2910  32 F5 64    LD (64F5),A   2ud
2913  CD E1 72    CALL 72E1     Mar
2916  CD 21 2A    CALL 2A21     M!*
```

```
4) Insider search for 3E,05,32
   >.2>.200FF01FF02FF03FF04FF05FF06FF07FF08FF09FF0AFF0BFF0CFF0DFF0EFF0FFF10FF1
1FF12FF13FF14FF15FF16FF17CB **>.2>.218CA19CA1ACA1BCA1CCA1DCA1ECA1FCA20CA21CA21D
5 **>.2>.222D423D424D425D42684 **>.2>.2270F **>.2>.2280E290E **>.2>.22A0D2B0D2C
0D2D0D2E0D2F0D300D310D320D330D340D350D360D370D380D390D3A0D3B0D3C0D3D0D3E0D3F0D4
00D410D420D430D440D450D460D470D480D490D4A0D4B0D4C0D4D0D4E0D4F0D500D510D520D530D
540D550D560D570D580D590D5A0D5B0D5C0D5D0D5E0D5F0D600D610D620D630D637D **>.2>.264
7C657C6674 **>.2>.
```

# ADVANCED BASIC

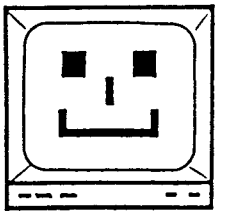
## USEFUL SUBROUTINES

This issue's section on Advanced BASIC contains two useful routines which have been sent in by readers. The first comes from D. G. Sherwood of Old Bilton near Rugby, and it asks the user to enter a number in either decimal, binary or hexadecimal form. The program then checks to see whether the number is within a given range and then converts it into the other two forms.

Any binary number needs to be preceded by an &X (this is standard notation for all binary numbers) and can be upto 16 digits long. All hexadecimal numbers must begin with an & sign, however, and can be upto 4 digits (0-F) long.

Thus, the routine can handle all sixteen bit numbers whether signed or not and is called by use of GOSUB 1000. Alternatively, if you just want the routine to convert a known number, rather than asking for one, just store the number in num$ and then GOSUB 1010. This subroutine could be used in conjunction with the inputting routines from last issue to provide a very polished number conversion program.

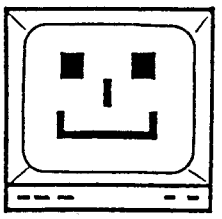
```
[37] 10 GOSUB 1000
[E4] 20 END
[BC] 1000 CLS:INPUT "Enter number to convert ",num$
[51] 1010 IF LEFT$(num$,1)="&" AND UPPER$(MID$(num$,2,1))<>"X" THEN GOSUB 1110:
       GOTO 1030
[AA] 1020 n=VAL(num$)
[C3] 1030 CLS:IF n THEN GOSUB 1050
[67] 1040 RETURN
[62] 1050 IF n<-32767 OR n>65536 THEN PRINT "NUMBER OUT OF RANGE":RETURN
[95] 1060 dc$=MID$(STR$(n),2):bn$=BIN$(n,1):hx$=HEX$(n,4)
[7D] 1070 PRINT TAB(20)"Binary  = &X";bn$
[B8] 1080 PRINT TAB(20)"Decimal = ";dc$
[1C] 1090 PRINT TAB(20)"Hex =     &";hx$
[5D] 1100 RETURN
[4E] 1110 num$=(MID$(num$,2,4))
[76] 1120 IF LEN(num$)<4 THEN num$=STRING$(4-LEN(num$),48)+num$
[C6] 1130 left=VAL("&"+LEFT$(num$,2)):right=VAL("&"+MID$(num$,3,2))
[63] 1140 n=256*left+right
[6C] 1150 RETURN
```



---

Sticking with the numbers theme, the second subroutine, which comes from Barrie Snell of Southsea, asks for a decimal number & then changes it into its written equivalent. For example, 567 becomes 'five hundred and sixty seven'. The actual subroutine begins at line 160 and the program includes a demonstration mode.

```
[C8] 10 REM Numbers to Words, Subroutine and Demonstration.
[BD] 20 REM   by Barrie Snell 1989.
[1C] 30 GOSUB 370:    ' Initialise
[F3] 40 MODE 2:t=17:n=t:GOSUB 160:PRINT n9:PRINT w$:PRINT
[F2] 50 FOR j=942 TO 1130 STEP 47
[C5] 60 n=j:GOSUB 160
[8A] 70 PRINT n9:PRINT w$:PRINT
[A5] 80 NEXT
[C4] 90 PRINT:PRINT"Enter a whole number from 0 to 999999999"
[FB] 100 PRINT:INPUT j
[E3] 110 IF j<0 OR j>999999999 OR j<>INT(j) THEN 90 ELSE n=j
[DC] 120 CLS:GOSUB 160
[88] 130 PRINT n9;:PRINT"is :-":PRINT:PRINT w$:PRINT
[D4] 140 GOTO 90
[AB] 150 '
[4E] 160 w$="":n9=n:GOSUB 320
[8C] 170 ON 3-INT((x-1)/3+0.1) GOSUB 190,230,250
[CB] 180 RETURN
[44] 190 n=FNf(x-6):a=FNm(x-5,3):b=FNr(3):GOSUB 250
[40] 200 w$=w$+"million "
[E6] 210 n=b
[B7] 220 IF a+b=0 THEN RETURN ELSE IF a=0 THEN IF b<100 THEN w$=w$+"and ":GOSUB
     250:RETURN ELSE GOSUB 250:RETURN ELSE n=a*1000+b
[B2] 230 GOSUB 320:n=FNf(x-3):a=FNr(3):GOSUB 250:w$=w$+"thousand "
[FB] 240 IF a=0 THEN RETURN ELSE n=a:IF a<100 THEN w$=w$+"and "
[81] 250 GOSUB 320
[A5] 260 IF x+r=1 THEN w$="zero ":RETURN ELSE ON 4-x GOSUB 270,290,310:RETURN
[48] 270 f=FNf(1):m=FNm(x-1,1):w$=w$+u$(f)+" hundred "
[4C] 280 IF m+r=0 THEN RETURN ELSE w$=w$+"and "
[66] 290 f=FNf(1):m=FNm(x-1,1)
[E9] 300 IF m=1 THEN w$=w$+tn$(r)+" ":RETURN ELSE IF m>1 THEN w$=w$+t$(m)+" "
[5D] 310 w$=w$+u$(r):IF r>0 THEN w$=w$+" ":RETURN ELSE RETURN
[13] 320 n$=RIGHT$(STR$(n),LEN(STR$(n))-1):x=LEN(n$):r=FNr(1):RETURN
[C7] 330 DATA one,two,three,four,five,six,seven,eight,nine,ten,eleven
[8C] 340 DATA twelve,thirteen,fourteen,fifteen,sixteen,seventeen
[6E] 350 DATA eighteen,nineteen,twenty,thirty,forty,fifty,sixty,seventy
[85] 360 DATA eighty,ninety
[9A] 370 RESTORE 330:FOR k=1 TO 9:READ u$(k):NEXT
[63] 380 FOR k=0 TO 9:READ tn$(k):NEXT
[32] 390 FOR k=2 TO 9:READ t$(k):NEXT
[0D] 400 u$(0)="":DEF FNf(a)=VAL(LEFT$(n$,a))
[6E] 410 DEF FNm(a,b)=VAL(MID$(n$,a,b))
[D1] 420 DEF FNr(a)=VAL(RIGHT$(n$,a))
[C4] 430 RETURN
```



All programs in Print-Out have Linecheck codes which are enclosed in brackets at the start of a line. Don't enter them in as they're designed to be used with Linechecker to eliminate errors when typing in programs which appear in this magazine. Please note, all programs will run whether Linechecker is being used or not. For information on how to use Linechecker, please see Issue Three.

It is envisaged that most users will want to include this as part of a far larger program (possibly as a score routine) & in this case it's important that you initialise the data at the very beginning of the program by using GOSUB 370. When you next want to convert a number all you need to do is, load the value to convert into 'n' and then GOSUB 160. On exit from the routine, w$ contains the value in words and n9 contains the actual number (in digits). Remember that you can RENUMber the subroutine to whatever line numbers suit you. As it stands the routine first prints out several example numbers, and then gives you the chance to enter numbers of your choice (it can only handle whole numbers between 0 and 999999999 but you should be able to modify it to print negative numbers, & even decimal places, if required) and it is then printed out, both in numerical form and in words. Below are a few programming notes which should be born in mind if you are using the routine inside another program.

The variable n is corrupted by the routine but its value is stored in n9. It is most important that the variable n must never be DEFined as an INTeger, & it must be restricted for use only in this subroutine. If you have a line, such as this....

```
800 p=p+1:LOCATE 2,7:PRINT "Score";p;"points":GOTO 400
```

....and you want to convert it to use this subroutine, you need to transfer 'p' into 'n' and then GOto the SUBroutine. Finally, note the extra space before w$, this is because w$ does not have a leading space like p does. The line would be

```
800 p=p+1:n=p:GOSUB 160:LOCATE 2,7:PRINT "Score ";w$;points":GOTO 400
```

That's it for this issue, but if you have any useful routines or any queries, then please contact us at the usual address.

---

# COMPETITION

We are feeling in a generous mood (in readiness for our birthday issue!!!) and have got a couple of prizes to give away to the first two people to send us the completed form below. The prizes are all of the program tapes or discs from the first six issues of Print-Out and they are very easy to win. All you've got to do is fill in the form below and tell us whether you want your prize on tape or disc. To give you loads of time to send your entries in, the closing date is October 30th and the winners will be printed in Issue 8, along with the winners of the competitions from Issue Seven. So, what have you got to lose ?? Send the completed form to: PRINT-OUT, 8 Maze Green Road, Bishop's Stortford, Herts.

---

NAME :..... To: PRINT-OUT
ADDRESS :....., 8 Maze Green Road
....., Bishop's Stortford
POSTCODE :..... Herts CM23 2PJ.

I own a CPC ..... and would like my prize sent to me on TAPE/DISC. (Delete as applicable).


# News & Views


It's been a hectic month as far as the big names have been concerned, with many software houses contemplating the possibilities of Amstrad's new computers and with many fighting to follow Ocean's lead in producing console based games. However, there have also been developments as far as the CPC is concerned which rarely get mentioned in the national computer magazines.

## Gaming Newsletter

One such item's a newsletter produced by Carl Surry for the gaming side of the CPC. His occasional 13 page plus newsletter costs just 30p and a large SAE (or 70p including postage and packing) and he has now produced his third issue. The articles are written in a lively format and feature many excellent examples of clip art from Stop Press. Some of the newsletter is given over to Carl's own thoughts and comments on various topics, and he also invites readers' own views and contributions. The rest of the pages include games and other reviews, tips, cheats and pokes (you will often see his name in Amstrad Action's 'Cheat Mode'). "Carl's Mini Newsletter" as it is called, is an excellent buy and copies can be obtained from Carl Surry, 37 Fairfield Way, Barnet, Herts EN5 2BQ.

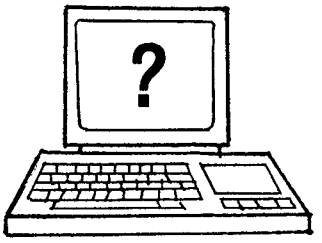
## Data PD

Tony Kingsmill, homebrew adventure writer, has now started up DATA PD that is the latest in an ever growing number of Public Domain libraries for the CPC. All the software is on disc and he produces a bi-monthly leaflet which includes a list of all the PD that's currently available. Copies of this can be obtained by sending 30p or an SAE. The cost of the actual PD is based on the number of K (Kilobytes) of code that you order - the price is 2p per K. Thus a full 2 sides of a CF-2 disc (356K) would cost £7.12 plus a 3" disc and an SAE for the return of your disc. One of the best points of this library is your ability to 'pick & mix' the programs thus only ordering what you really want. Further details are available from Tony Kingsmill,DATA PD Library,202 Park Street Lane,Park Street, St Albans,Herts AL2 2AQ.

## SORRY!! Homebrew section full

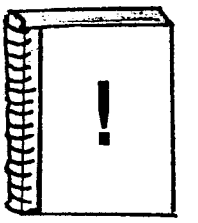
We have received so many homebrew games to be reviewed in this issue, that we have been unable to squeeze them all in (hence the rather brief reviews this issue - so we could include as many as possible). But don't worry, those of you who sent in programs which did not get printed this time round, will be printed in the next issue of Print-Out. If you have something you wish to publicise, or have comments about anything to do with the CPCs, then please write to us at :

PRINT-OUT, 8 Maze Green Road, Bishop's Stortford, Hertfordshire CM23 2PJ.



# TECHNICAL TIPS

by Bob Taylor



## INTERRUPTS in M/C

Two readers have sent in queries regarding interrupts in Machine Code and as my replies are quite long, I won't have room to explain about interrupts at any length - perhaps that could be done in a separate article in the future.

The first query is for a routine 'to scroll letters in the bottom five to eight lines of the screen under interrupt'. The request immediately threw up a number of questions about the reader's actual requirements:

1 Which Mode?
2 How large are the letters?
3 On how many lines are they to be scrolled and if so, how?
4 How many colours?

I chose the easy way out & opted for Mode 2 with scrolling standard height and width characters on one line only. The routine on the next page is pared to a bare minimum.Even so, out of every 20ms of computing time, the Event Handling part of it takes at least 3.6ms and this cannot be spent running the foreground program which, therefore, runs slower. (At the expense of an extra 154 bytes of room, this time could be reduced to 2.8ms per line scrolled by substituting 79 LDI instructions in place of LD C,&4F: LDIR). To calculate for several lines of scrolling, multiply up the interrupt time proportionately; this might result in impinging upon the 'screen printing area scanning time' and if enough lines are used, no time at all may be left to run the foreground program. Choosing to use big characters causes a similar multiplication depending on the number of lines they cover. Using Modes 0 or 1 or colours only slightly increases the interrupt handling time but adds to the length of the routine. The routine once installed is initiated by the syntax: CALL &8000,<string to be scrolled>.

---

You will notice that we have included only the disassembly in this issue, as we envisage most users modifying it for their specific purpose. However, if you do want a version to run from BASIC, one is included on this issue's program tape/ disc.When the Machine Code has been installed, the following short program will repeat the scrolling until it is halted by pressing [ESC]; pressing [ESC] again will exit to BASIC:

```
[52] 10 MODE 2:EVERY 100 GOSUB 30
[7C] 20 GOTO 20
[B2] 30 CALL &8000,"Hello Hello HELLO":RETURN
```

NB: The time after EVERY must allow at least 1/50 Sec for each character of the string plus another 81/50 Sec to scroll it off the screen. NB: Also that if the screen itself has been scrolled it will not be possible to predict the position of the line to be scrolled and this may lead to some strange effects.

The disassembly of the message scroller:

```
ORG  &8000
.init
DEC  A
RET  NZ          ;do nothing if not 1 parameter (for the String)
LD   HL,lenrem   ;HL-> Length Remaining
LD   A,(DE)      ;get the String Length from the String Descriptor:
                 ;on entry to this routine, DE holds the last para-
                 ;meter (the address of the String Descriptor)
LD   C,A         ; into C also
ADD  A,&51       ;allow for moving String completely off Screen
LD   (HL),A      ; store in lenrem (Length Remaining)
LD   A,C         :get Length again
INC  HL          ;HL-> start of String Buffer
LD   (strpos),HL ;store address of Buffer in String Position
LD   (HL),&20    ;place Space at start of String
EX   DE,HL       ;DE-> String Buffer; HL-> String Length in Descriptor
INC  HL          ;step on to the address LB in the String Descriptor
LD   B,(HL)      ;get address of string into HL
INC  HL          ;      "          "
LD   H,(HL)      ;      "          "
LD   L,B         ;      "          "
LD   B,&00       ;BC= String Length
INC  DE          ;DE-> byte after Space (Buffer destination for String)
LDIR             ;transfer String to Buffer
XOR  A           ;
LD   (DE),A      ;signal 'End of String' with &00 byte
LD   B,&81       ;Event Class: Asynchronous, with routine in RAM
LD   DE,evthnd   ;address of Event Handling routine in RAM
LD   HL,evtblk   ;address of 9 byte Event Block
JP   &BCD7       ;KL NEW FRAMEFLY to initialise the Event
.evthnd
PUSH HL          ;-> (1) HL: save all registers
PUSH DE          :-> (2) DE    "
PUSH BC          :-> (3) BC    "
PUSH AF          :-> (4) AF    "
CALL &B903       ;KL U ROM DISABLE: sometimes results in garbage if
                 ; omitted
PUSH AF          ;-> (5) original ROM Selection state
CALL &B906       ;KL L ROM ENABLE: to allow access to Character Matrix
LD   HL,(strpos)
LD   A,(HL)      ;get 1st/next character of String
OR   A           ;Zero flag set if the 'End of String' byte
JR   Z,pshhl     ;if at end of String, don't calculate Matrix address
INC  HL          ;otherwise step strpos on for next interrupt
LD   (strpos),HL ;      "          "          "
LD   H,&07       ; and calculate address of relevant Character Matrix
LD   L,A         ;       "
ADD  HL,HL       ;       "
ADD  HL,HL       ;       "
ADD  HL,HL       ;       "
.pshhl
PUSH HL          ;-> (6) Char Matrix
LD   B,&00       ;pre-load B with &00 outside 'row' loop (for the LDIR)
LD   HL,&C730    ;HL-> start of Screen line 24 top row; change to
                 ; address of start of other Lines if required
.nxtrow
LD   E,L         ;
LD   D,H         ;DE-> start of this line/row also
INC  HL          ;HL-> 2nd byte in this row
LD   C,&4F       ;BC= 79 bytes to move
LDIR             ;move last 79 bytes of row, one byte to the left
JR   Z,stobyt    ;if end of String, use &00 byte to clear Screen byte
POP  HL          ;otherwise use Char Matrix byte (5)
LD   A,(HL)      ;get 1st/next byte from Matrix
INC  HL          ;step on to next byte for next row
PUSH HL          ;-> (6) Char Matrix
.stobyt
LD   (DE),A      ;load Screen RAM with Matrix byte or clear with &00
LD   HL,&07B1    ;
ADD  HL,DE       ;HL-> start of next row of this Screen line; Carry
                 ; flag set only if last (8th) row of line just done
JR   NC,nxtrow   ;if more rows to do
POP  HL          ;discard Char Matrix (5)
POP  AF          ;original ROM selection state (4)
CALL &B90C       ;KL ROM RESTORE to reset ROMs to their entry states
LD   HL,lenrem   ;(Length Remaining)
DEC  (HL)        ;decrement count of scrolls left to do
LD   HL,evtblk   ;(Event Block)
CALL Z,&BCDD     ;KL DEL FRAME FLY to turn off the interrupt if no more
                 ; scrolls
POP  AF          ;AF (3)
POP  BC          ;BC (2)
POP  DE          ;DE (1)
POP  HL          ;HL (0)
RETI             ;Return from the interrupt to the original routine
.evtblk DEFS 9          ;reserve 9 bytes for the Event Block
.strpos DEFW &          ;address of the next char to print on the Scree
.lenrem DEFB &          ;count of number of scrolls still required
.strbuf DEFB " ",(string to be printed),&00  ;String to be scrolled with
                        ;leading Space and 'End of String' byte
```

# The 5th Interrupt

The second query concerns reducing flickering on large sprites. Mr Messina of Heywood had read in Amstrad Action some time ago about using the '5th Interrupt' in order to give more time to process a large sprite.After having re-read The Look on page 28 of the February 1989 issue, I can understand his difficulty. The '5th Interrupt' referred to is actually just one out of the six Fast Ticker interrupts which occur for every Frame Flyback.The Fast Ticker interrupts occur every 1/300 Sec whereas the Frame Flyback one has a period of 1/50 Sec.

Normally, a screen routine running under interrupt, uses the Frame Flyback one (as in the previous routine) which gives approx 1/300 Sec in which to carry out its task.This 1/300 Sec is the time taken for the raster to travel from the very top of the Screen down to the top of the printing area. During this time, of course, nothing in the printing area is being scanned, so no flickering will result. However if the routine requires longer than this, the screen will still be being addressed while the raster's drawing out the changes & the result will be flickering caused by pixels being changed at the very moment the raster spot moves over them.

If, instead of the Frame Flyback interrupt, we use the correct Fast Ticker one, we can find ourselves with 2/300 Sec to work in. The six Fast Ticker interrupts start with one at the very top of the Screen (which we number 0). Number 1 occurs at the start of the printing area; Nos 2, 3 and 4 occur during the raster scanning of the printing area. We should not alter the Screen while the printing area is being scanned, so we cannot use Nos 1, 2, 3 or 4 to trigger our routine. The 5th - the one we are after - is just at the end of this area. If we use this one, we have 1/300 Sec while the raster scans the lower non-printing area, plus the 1/300 we had before, so doubling the time available for our routine.

---

In order to pick out the one we need, we have to keep a count of these six interrupts. We do this, by initialising a counter with a value of 5, which will be decremented as each interrupt occurs. When this value is reduced to 0, we're at the '5th interrupt' that we want (it is then re-loaded with a 6 for the next count cycle). The sprite moving routine (without its previous interrupt handling section) should follow on from the end:

```
.init   LD   HL,evtblk   ;address of 9 byte work area for the Operating System
        OR   A           ;check number of parameters - none to disable the
                         ; interrupt
        JP   Z,&BCE6     :KL DEL FAST TICKER to disable
        LD   B,&03       ;  Call MC WAIT FLYBACK several times to synchronise
.nxtfrm CALL &BD19       ;   the sequence of Fast Ticker interrupts which are
        DJNZ nxtfrm      ;   to follow
        LD   A,&05       ;
        LD   (intctr),A  ;prime the Interrupt Counter with 5
        LD   B,&81       ;Event Class: Asynchronous, with routine in RAM
        LD   DE,evthnd   ;DE-> Event Handling routine
        JP   &BCE0       ;KL NEW FAST TICKER to initialise
```

```
.evtblk DEFS 9          ;reserve 9 bytes for Event Block
.intctr DEFB &          ;Interrupt Counter
.evthnd PUSH HL         ;-> (1) HL: save registers
        PUSH AF         ;-> (2) AF:   "
        LD   HL,intctr  ;HL-> Interrupt Counter
        DEC  (HL)       ;decrement Interrupt Counter
        JR   Z,resctr   ;if on '5th interrupt' then reset Interrupt Counter -
                        ; and continue with sprite handling
        POP  AF         ;AF (1)
        POP  HL         ;HL (0)
        RETI            ; otherwise ignore this interrupt (not the 5th)
.resctr LD   (HL),&06   ;set Interrupt Counter ready to find next '5th'
        PUSH DE         ;-> (3): save registers
        PUSH BC         ;-> (4):   "
.sprite                 ;the sprite handling routine should follow on here
```

I have assumed that the routines will run in RAM;the Interrupt Counter has to be in RAM and the Event Block must be in the central 32K of RAM.

To enable the Event from BASIC,the initialisation routine should be CALLed with a dummy parameter (anything will do). If starting up within a M/C routine, the A register should contain anything but 0 when 'CALL init' is used. In order to disable the Event,no parameter should accompany a CALL to the initialisation routine (for M/C the A register should contain 0).

The Operating System tolerates multiple enabling or disabling of the Event, although this should not occur with good programming.

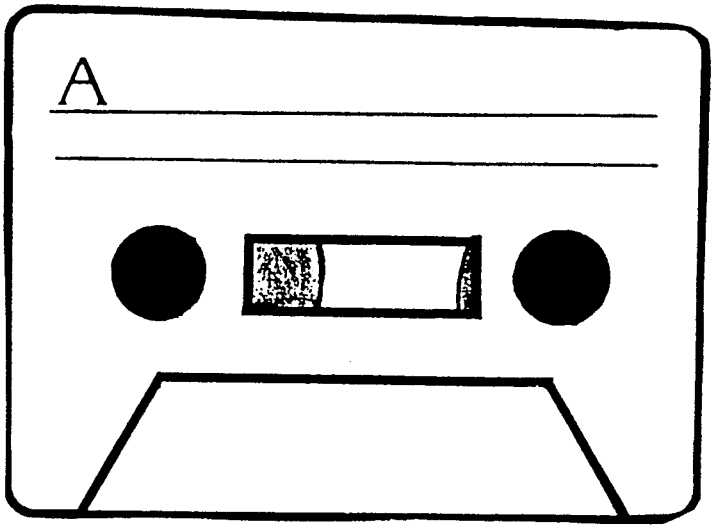
---

If more than one sprite is required (although given the problems Mr Messina was having with flickering due to lack of time, I don't think there will be much time available for more than one), as many sprites as you need could be serviced in the one sprite routine following on Event Handling;but as an alternative, the Initialisation, Event Block, Interrupt Counter and Event Handling routines/areas could be duplicated for each sprite. With a number of separate Events, the sequence in which each is enabled will usually determine the order in which they are serviced. However if a pre-determined sequence is required, a priority value can be included in the Class value, loaded into the B register, just before &BCE0 is JumPed to.There are 16 priorities available numbered 0 to 15 with 0 as the highest (first); whichever number is chosen, it should be doubled (giving a range of &00, &02, &04 etc up to &1C, &1E) and this should be ADDed or ORed to the value indicated for the B register.

As Fast Ticker interrupts must be serviced 6 times for every Frame Flyback one, their use will slow the running of the main program a little more. There's nothing that can be done about this, other than to make both the Main Routine & the Sprite routine as fast as possible when writing them; if Frame Flyback interrupts allow enough time for your routines then use them in preference to Fast Ticker ones.

---

# Offers




Please make all cheques payable to Print-Out but any postal orders should be made out to T J Defoe as this saves the Post Office a great deal of time and effort. Unless it cannot be avoided, it is advisable not to send cash through the post. All orders should be sent to :- PRINT-OUT, Special Offers, 8 Maze Green Road, Bishop's Stortford, Hertfordshire CM23 2PJ.

## *ISSUE SEVEN*

If you wish to order a copy of Issue Seven in advance you may do so by sending a cheque / postal order for £1.10 (or 70p + an A4 SAE with a 28p stamp) to the usual address. We hope to have it published by about the 30th September and as soon as it is printed it will be forwarded to you.

## *PROGRAM TAPES & DISCS*

We now supply both program tapes and discs for ALL issues and the prices given below also include a booklet to explain how the programs work plus postage and packing. Tapes & discs are available for Issues One,Two,Three,Four,Five & Six.
The cost for a program tape is as follows :-

a) A blank tape (at least 15 minutes) and 50p (p+p)
or b) £1.00 (which also includes the price of a tape)

The cost for a program disc is :-

a) A blank formatted disc and 50p (p+p)
or b) £3.00 (which also includes the cost of a MAXELL/AMSOFT disc) *

* When ordering using this particular method please allow about 21 days for delivery as we must rely on outside suppliers for the discs.
* Please also note that one side of one CF-2 disc will hold upto six issues worth of program discs (ie. 1 disc holds all the programs from 12 issues)

## *BACK ORDERS*

We still have some copies of Issues 1,2,3,4 and 5 available and the price is £1.10 which includes postage and packing. Alternatively you can order both a back issue and its corresponding tape or disc by sending :-

a) £1.75 - includes the tape, the required issue and postage and packing
b) £3.75 - includes the disc (genuine MAXELL/AMSOFT disc) & the required issue and postage and packing.

Send to: Print-Out, 8 Maze Green Road,
Bishop's Stortford, Herts.

NAME (block capitals please) .....
ADDRESS .....
.....
.....
POSTCODE .....

Please send me the following items :-

| DESCRIPTION | ISSUE NUMBER | QUANTITY | PRICE EACH | PRICE |
|---|---|---|---|---|
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | TOTAL PRICE | |

I enclose a cheque/postal order/cash* to the value of £..... Please make all cheques payable to PRINT-OUT and make postal orders out to Thomas Defoe. Thank you. (*delete as applicable)

# Subscription

If you are interested in having a subscription to Print-Out you will be glad to know that full details concerning subscriptions are printed on the next page. We are running two types of subscription - half-yearly (three issues) and also yearly (six issues) at the prices of £3.30 and £6.60 respectively.

# Advertising

You can place a small ad of upto 40 words (including your name and address) in this section of the magazine free of charge. If you wish to place a larger advertisement in the magazine, please write to us for a full list of advertising rates.

# Subscription Offer

Due to the many enquiries that we have received concerning subscriptions we have now introduced a subscription service and it will be operating from Issue Five. There are two forms of subscription :-

a) Three issues - approxiamately half a year

b) Six issues - approxiamately a full year

Although we do try and produce one magazine every two months this is not always possible due to other outside engagements and therefore exact release dates are not given in the magazine. Because of this, we are unable to guarantee that six issues will be produced in a year, or three issues in half a year. However, for a year's subscription you will be sent six issues no matter when they are published, and the same applies to a half-yearly subscription. If we stop producing the magazine, we promise to refund the cost of all unmailed issues.

The prices for subscriptions to Print-Out are as follows :-

| NO OF ISSUES | UNITED KINGDOM | EUROPE | REST OF THE WORLD |
|---|---|---|---|
| SINGLE | £1.10 | £1.50 | £2.00 / £2.50* |
| THREE ISSUES | £3.30 | £4.50 | £6.00 / £7.50* |
| SIX ISSUES | £6.60 | £9.00 | £12.00 / £15.00* |

* The first price quoted is for 'Printed Paper Rate' but this does not have the same level of security as a normal letter or parcel. The second price mentioned is for normal rate and is sent in the same way as an ordinary letter or parcel. For the United Kingdom there is an alternative price for ordering only a single issue and this is:- 70p + a large A4 SAE (with 28p stamp). We try to despatch all orders within a week of receiving them & all items are sent by SECOND CLASS post, unless the extra money or stamps are sent with your order.

---

## *Subscriptions*

Send to: Print-Out, 8 Maze Green Road,
Bishop's Stortford, Herts.

NAME (block capitals please) .....

ADDRESS .....

.....

.....

POSTCODE .....

Please send me the next three/six* issues of Print-Out as soon as they are published. I enclose a cheque/postal order* to the value of £..... and I wish my subscription to start from Issue ..... (* delete as applicable)
Please make CHEQUES payable to PRINT-OUT and make postal orders payable to Thomas Defoe (as this saves time and effort for the Post Office !!!).